Computerlinguistik und Künstliche Intelligenz

Natürlichsprachliche Dialoggesteuerte Datenbankabfrage

MAGISTERARBEIT
ZUR ERLANGUNG DES
MAGISTER ARTIUM

Im Fachbereich
Sprach- und Literaturwissenschaft
der Universität Osnabrück

vorgelegt von:
Rüdiger Rolf
aus Osnabrück
2002

Inhaltsverzeichnis

1 Einleitung.....7
2 Grundlagen.....11
2.1 Datenbanken.....11
2.2 Natürliche Sprache.....12
2.2.1 Vor- und Nachteile natürlicher Sprache.....13
2.2.2 Analyse des Satzes.....17
2.2.2.1 Syntaktische und semantische Analyse.....18
2.2.2.2 ATN-Grammatiken.....21
2.2.2.3 Mustererkennung.....21
2.2.3 Linguistische Phänomene.....22
2.2.3.1 Anaphorische Referenz[6].....23
2.2.3.2 Ellipsen.....24
2.2.3.3 Sprechakte.....25
2.2.3.4 Kontext.....26
2.2.3.5 Unterschiedliche Bedeutungen von „Und“[7].....26
2.2.3.6 Weitere Phänomene.....28
2.2.4 Kontextrepräsentation.....30
2.2.4.1 Diskursrepräsentationstheorie[8].....30
2.2.5 Phonetische Namenssuche.....32
2.3 Generieren von Antworten und Fragen.....32
2.4 Dialoge.....37
2.4.1 Ergänzung der Eingabe.....37
2.4.2 Turn Taking.....38
2.4.3 Shift of Intention.....39
2.4.4 Antizipation der Antwortmöglichkeiten.....40
2.4.5 Strategie der Dialogführung.....42
2.5 Besonderheiten natürlichsprachlicher maschineller Schnittstellen.....44
2.5.1 Teilmenge der natürlichen Sprache.....44
2.5.2 Unterschiede zwischen gesprochener und geschriebener Sprache.....45
2.5.3 Computertalk.....49
2.5.4 Weitere Unterschiede zur Kommunikation zwischen Menschen.....51

2.5.5 Dialogsprache....52
2.6 Auswirkung der im Telefonauskunftsprogramm gewählten Benutzerschnittstelle....52
3 Überblick über existierende Systeme....54
3.1 User Specialty Languages....54
3.2 Q&A....56
3.3 Ask Jeeves....57
3.4 C@ll + Find....59
3.5 COMODA....60
3.6 DICOS....62
3.7 Schlussfolgerungen....65
4 Beschreibung des implementierten Systems „Telefonauskunft“....67
4.1 Vorhandene Funktionalität....68
4.1.1 Der Client....68
4.1.2 Die Grammatik....68
4.1.3 Das Lexikon....69
4.1.4 Systemverhalten....69
4.1.5 Suchoptionen....70
4.1.6 Steuerbarkeit....70
4.1.7 Die Dialogkomponente....71
4.1.8 Fehleranalyse....71
4.1.9 Fehlertoleranz....72
4.2 Architektur....72
4.2.1 Das Benutzerinterface....73
4.2.2 Der Parser....75
4.2.3 Das Datenobjekt....81
4.2.4 Analyse der Datenbankabfrageergebnisse....81
4.2.5 Generieren der Systemreaktion....82
4.3 Aufbau der Datenbank....83
4.4 Ablauf des Programmes....86
4.5 Protokollierung....87
4.6 Verbesserungen und Erweiterungen für das Programm....87
4.6.1 Erweiterung und Verbesserung des Datenbestandes....88
4.6.2 Verbesserung der Korrekturmöglichkeiten....89
4.6.3 Erweitern der syntaktischen und semantischen Funktionalität....90

4.6.4 Verbesserung der Datenaufbereitung....91
4.6.5 Verbesserung der Dialogkomponente 91
4.6.6 Verbesserung der Textgenerierung....92
4.6.7 Verbesserte Erfassung des Benutzerverhaltens....92
4.6.8 Benutzermodellierung....92
4.6.9 Verbesserung und Erweiterung der Benutzerschnittstelle....93
4.6.10 Datenschutz....94
4.6.11 Automatisches Testen des Programms....94
4.7 Zusätzliche Programme 94
4.7.1 Der Generator....95
4.7.1.1 Person erstellen....95
4.7.1.2 Firma erstellen....96
4.7.1.3 Abteilung erstellen....96
4.7.1.4 „Arbeitsamt“....97
4.7.1.5 Ort erstellen....97
4.7.1.6 Gewerbe erstellen....97
4.7.2 Die konventionelle Schnittstelle....98
4.8 Technisches....99
5 Ergebnis der Evaluation....101
5.1 Deskriptive Merkmale....102
5.2 Ergebnisse aus der Videokonfrontation....105
5.2.1 Datenausgabe....106
5.2.1.1 Feedback....106
5.2.1.2 Dialogqualität....106
5.2.1.3 Sprachverständnis....107
5.2.1.4 Ausgabelogik....107
5.2.2 Designaspekte....108
5.2.3 Eingabeverarbeitung....108
5.2.3.1 Eingabeart....108
5.2.3.2 Handlungsstrategie....109
5.2.3.3 Eingabeform....109
5.3 Bewertung des Systems durch die Benutzer....109
5.4 Schlussfolgerungen....113
6 Diskussion....115

6.1 Ausblick.....121
7 Literaturverzeichnis.....123

1 Einleitung

In dieser Arbeit wird gezeigt, wie natürlichsprachliche (NL) Schnittstellen für Computer und insbesondere Datenbanken durch die Möglichkeit wechselseitiger aufeinander aufbauender und bezugnehmender Äußerungen ergänzt werden können.

Diese Art des Dialoges ist einen weiteren Schritt von natürlichsprachlichen Schnittstellen hin zu natürlichen Schnittstellen. Damit soll nicht gemeint sein, dass die Kommunikation an einer solchen Schnittstelle durch diese Änderung der Kommunikation zwischen Menschen gleichkommt, dazu ist die Verarbeitung der natürlichen Sprache durch den Computer noch nicht weit genug fortgeschritten, aber die Einschränkung, dass eine Anfrage an eine Datenbank in *einer* Aussage formuliert werden muss, entfällt hier. Die Notwendigkeit mehrere Aussagen in einer zusammenzufassen, ist eine Denkweise, die erst erlernt werden muss, auch wenn sich viele Anwender inzwischen wahrscheinlich sehr daran gewöhnt haben.

In einer natürlichen Kommunikation steht es dem Fragenden zu einem späteren Zeitpunkt immer noch frei, seine Frage zu verfeinern oder sogar zu verändern und mit dem Gefragten kooperativ eine gemeinsame Wissensgrundlage zu schaffen, welche es ermöglicht, die Frage zu beantworten.

Die Gricesche Maxime der Quantität[1] (Grice, 1989) zu erfüllen ist nicht immer leicht. In einem wechselseitigen Dialog würde der eine Gesprächsteilnehmer den anderen auf das Fehlen benötigter Informationen aufmerksam machen. Wenn der Redebeitrag aber aus nur einer Äußerung bestehen darf und als Reaktion nur eine Fehlermeldung oder ein Ergebnis möglich ist, muss diese Maxime aber hunderprozentig umgesetzt werden.

Natürlich sollte eine natürlichsprachlichen Schnittstelle auch die anderen Griceschen Maximen der Qualität, Relation und Art und Weise (Grice, 1989) erfüllen. Dies gilt insbesondere auch für die vom Computer erzeugten Teile der Kommunikation.

Viele Systeme versuchen, aufeinanderaufbauende Anfragen zu verhindern, so z.B. auch das Bibliothekssystem OSIRIS, das nach jeder Anfrage wieder den Eingangsbildschirm präsentiert um den Eindruck zu erwecken, der Benutzer begegne dem System das erste Mal. (Ronthaler, 2000, Kapitel 4.2.3)

Dialog ist, im Bereich von Computer betrachtet, ein ziemlich weit gefasster Begriff. Ein Dialog kann ein Fenster mit einem Knopf oder ein Texteingabefeld auf einer Internetseite sein,

1 Mache deinen Beitrag so informativ wie in der momentanen Situation benötigt und mache deinen Beitrag nicht informativer als nötigt.

bei der der Computer irgendwie auf die Eingabe reagiert. Schlüssiger scheint der Begriff Dialog zu sein, wenn man es mit mehreren Feldern in einem Formular zu tun hat, wo für jedes Feld eine Eingabe zu einem bestimmten Bereich verlangt wird, z.B. Name, Straße und Ort. Dies kann man vielleicht als eine Methapher für eine Frage sehen und die darauf erwartete Antwort.

Etwas, dass man umgangssprachlich als Dialog bezeichnet, findet man, wenn man sich Systeme anguckt, die gesprochene Sprache verstehen, auch wenn der Dialog hier zumeist doch nur eine sprachliche Umsetzung der eben erwähnten Formulare ist. Im Dialogzug des Computers wird eine Frage gestellt und im Zug des Benutzers wird eine Antwort erwartet. Die Initiative liegt immer beim Computer. Eine Wechsel in der Initiative findet nicht statt, auch wenn es sich anbieten würde, um den Dialog effizienter zu gestalten.

Die sprachlichen Fähigkeiten natürlichsprachlicher Systeme sind begrenzt. Durch eine Dialogkomponente kann ein System möglicherweise selbst seine Grenzen aufzeigen und sagen, welche Worte oder Formulierungen es nicht versteht. Es kann den Benutzer langsam an die Einschränkungen des Systems heranführen. In einigen natürlichsprachlichen Systemen gibt es Einschränkungen oder vordefinierte Phrasen über die man sich erst informieren muss, bevor man das System benutzen kann. Der Vorteil, dass natürlichen Sprache nicht erlernt werden muss, geht dabei verloren.

Durch eine geschickte Formulierung der Fragen des Systems kann der Eingabeaufwand an NL-Schnittstellen möglicherweise reduziert werden. Es ist nicht zu erwarten, dass ein Benutzer, vor allem wenn er tippen muss, daran interessiert ist, ganze Sätze zu benutzen, auch wenn dies manchmal wünschenswert wäre.

Der Schwerpunkt dieser Arbeit liegt nicht im Bereich Datenbanken und ihren Fähigkeiten. Datenbanken sind als Teil des Themas gewählt worden, weil sie ein Anwendungsgebiet sind, in dem NL-Schnittstellen einen einfachen und effizienten Zugriff erlauben sollten. Es geht hier darum, ob und wie es Benutzern möglich ist, möglichst einfach und ohne großen Lernaufwand Datenbanken abzufragen.

Ein normaler Benutzer ist sich nicht über die Möglichkeiten von formalen Sprachen zur Datenbankabfrage im Klaren, aber es sollte für ihn auch möglich sein, ohne dass er Befehle lernen oder die Struktur der Datenbank kennen muss, die von ihm gesuchten Daten zu finden.

Dass Benutzer Probleme mit formalen Schnittstellen haben zeigen z.B. die Untersuchung zur Benutzerschnittstelle des OPAC-Bibliothekssystems (siehe auch Gattung, 1991; Moranz, 1998; Peters, 1989). Die Anwender gehen normalerweise nach dem selben einfachen Schema vor, erweiterte Funktionen werden nur von Wenigen überhaupt genutzt, auch wenn sie die Suche vereinfachen würden.

Dadurch, dass die Struktur der Datenbank bei natürlichsprachlichen Schnittstellen weitestgehend verborgen wird, kann der Benutzer nicht wissen, wonach er fragen kann, bzw. welche Informationen er angeben muss um die gewünschte Antwort zu erhalten. Der Dialog kann dieses Manko beheben, indem der Benutzer gezielt nach den fehlenden Informationen gefragt wird. Im Weiteren ist es vorstellbar, dass das System gefragt werden kann, welche Informationen es zur Verfügung stellt, wie man einen Gesprächspartner z.B. auch fragen kann ob er weiß, wo die Johannisstraße ist.

Der Bedarf an Datenbankschnittstellen, die auch von Benutzern ohne besondere Vorkenntnisse zu bedienen sind, ist sicherlich groß. Alleine das Internet bietet Informationen ihn großer Menge, die für viele Benutzer teilweise nur schwer zugänglich sind, z.B. Versandhauskataloge, Lexika oder Immobilienangebote. Durch die schnell wachsende Popularität des Mediums Internet haben viele der neuen Benutzer nur ganz grundlegendes Wissen.

So ist es zum Besipiel auch bei Internetsuchmaschinen, mit der man Internetseiten zu einem speziellen Thema suchen kann. In den meisten Fällen lassen sich diese Datenbanken nur über Stichworte abfragen und liefern alle Seiten als Ergebnis, die zu dieser Anfrage passen.

Wünschenswerter wäre es natürlich, wenn ein System die Ergebnisse kategorisieren oder anderweitig ordnen würde und dann nachfragen könnte, welche Art von Dokument gesucht wird.

Ein Ziel dieses Ansatzes ist es, die Qualität der Suchergebnisse zu steigern, anstatt eine große Anzahl von Ergebnissen zu liefern. Bei den Internetsuchmaschinen zum Beispiel ist es dem Benutzer überlassen, aus der meistens großen Anzahl von Ergebnissen selber das passende zu suchen, er kann meistens nur weitere Stichworte angeben, um die Menge der Ergebnisse einzuschränken.

Im Rahmen dieser Arbeit ist ein Programm für eine Telefonauskunft entstanden, dass nach einer initiativen Frage eines Benutzers diesem so lange Nachfragen zu noch fehlenden

Informationen stellt, bis die Menge der Ergebnisse auf ein Minimum reduziert wurde. Dem Benutzer steht es dabei jederzeit frei, den Dialog abzubrechen und sich alle zu diesem Zeitpunkt gefundenen Ergebnisse anzeigen zu lassen. Das Programm wurde von Jens Schroeder und Sarah Helmich evaluiert. Einige der Ergebnisse dieser Evaluation werden in dem entsprechenden Kapitel noch vorgestellt.

2 Grundlagen

In diesem Abschnitt wird zunächst auf Datenbanken und Datenbankabfragemöglichkeiten eingegangen. Im weiteren werden die Techniken vorgestellt, die benötigt werden um natürliche Sprache maschinell zu verarbeiten und so zumindest in Teilen die Vorteile der natürlichen Sprachen zu nutzen. Zuerst einmal werden dazu die Vor- und Nachteile der natürlichen Sprachen an maschinellen Schnittstellen gezeigt. Danach werden Techniken vorgestellt, wie Sätze analysiert werden können und im Anschluss daran die linguistische Phänomene aufgezeigt die auftreten können. Ein weiteres Kapitel beschäftigt sich mit der Repräsentation des Kontexts. Den Abschluss dieses Teils bilden Informationen über die phonetischen Namenssuche.

Danach wird auf die Textgenerierung eingegangen, genauer gesagt was bei der Generierung von Fragen und Antworten durch den Computer zu beachten ist.

Es werden im weiteren Grundlagen zu Dialogen und Gestaltungsrichtlinien für diese gezeigt.

In einem weiteren Kapitel des Grundlagenteils werden Besonderheiten von natürlichsprachlichen Schnittstellen aufgezeigt. Der Schwerpunkt in diesem Teil hierbei liegt auf der vom Benutzer verwendeten Sprache.

Zum Schluss dieses Abschnitts wird noch eine Besonderheiten der Benutzerschnittstelle des Telefonauskunftsprogrammes aufgezeigt.

Vom Ansatz dieser Arbeit her gibt es bei vielen dieser Punkte unterschiedliche Herangehensweisen. Zum einen die sprachwissenschaftliche, die sich mit dem Aufbau und der Analyse der Sprache beschäftigt. Dann die Vorgehensweise der Informatik, welche die Verarbeitung der Information zum Ziel hat und eine Einschränkung auf eine kleinere, besser zu verarbeitende Teilmenge der gesamten Information nahe legt. Und zuletzt die psychologische, die sich um die Verständlichkeit des Vorgehens für den Benutzer und die Effizienz im Umgang mit der Software sorgt und somit eine Reihe von Richtlinien und Ansätzen für die Gestaltung bietet.

2.1 Datenbanken

In dieser Arbeit soll nicht auf die Fähigkeiten und den Aufbau von Datenbanken eingegangen werden. Trotzdem gibt es einige Grundlagen zu Datenbanken, die erwähnt werden müssen.

Es existieren eine Vielzahl von formalen Datenbankabfragesprachen unter denen SQL aufgrund seiner Verbreitung und Akzeptanz den Standard darstellt. SQL wurde 1979 zum ersten mal von Oracle in einer kommerziellen Datenbank eingesetzt[2].

Die Daten in einer relationalen Datenbank sind in Tabellen organisiert. Tabellen bestehen aus Zeilen und Spalten. Eine Zeile entspricht einem Eintrag. Eine Spalte enthält Werte eines Typs. Diese Typen können z.B. Text, Zahlen, das Datum und vieles mehr sein. Über die Werte der einzelnen Felder können Funktionen ausgeführt werden, wie das Vergeben einer eindeutigen Bezeichnung für einen Eintrag, durch ein automatisches Einfügen des nächstgrößeren Wertes in einer Spalte.

Bei der Suche in Datenbanken sind auch Verknüpfungen von Spalten unterschiedlicher Tabellen möglich. Dazu wird unter SQL der JOIN-Befehl verwendet.

Natürlich kann man das Ergebnis einer Suche nach einer oder mehreren Spalten sortieren lassen und die Spalten, die ausgegeben werden sollen, angeben.

SQL kann auch bei einigen einfachen natürlichsprachlichen Anfragen schon sehr komplizierte Ausdrücke benötigen, wie dieses Beispiel von Zoeppritz (1983) zeigt:
Der Satz „Welcher Mitarbeiter wohnt nicht in Heidelberg?“ entspricht dem SQL-Ausdruck:

```
SELECT UNIQUE X.MITARBEITER
      FROM EMP X
      WHERE X.MITARBEITER NOT IN
            (SELECT UNIQUE Y.MITARBEITER
                  FROM EMP
                  WHERE Y.WOHNORT='HEIDELBERG');
```

In diesem kleinen Überblick ist, die in dieser Arbeit benötigte Funktionalität von Datenbanken dargestellt.

2.2 Natürliche Sprache

Wenn von natürlicher Sprache in Bezug auf Computer die Rede ist, sollte einem bewusst sein, dass es sich hierbei nur um eine Teilmenge der natürlichen Sprache handelt. Die natürliche Sprache ist zu komplex um von heutigen Maschinen umfassend verarbeitet zu werden. Meistens wird aber der Informationsgehalt, den die natürliche Sprache bietet, nicht benötigt, um die Sprache in der Maschine erfolgreich verarbeiten zu können. Die natürliche Sprache enthält auch Redundanzen, die ignoriert werden können, aber unter Umständen die Analyse des Satzes aber auch komplizierter machen.

2 http://webopedia.internet.com/TERM/S/SQL.html

Die Reduzierung der natürlichen Sprache geschieht in der erster Linie durch die Verkleinerung des benutzbaren Vokabulars und auch durch Einschränkung der zu gebrauchenden Syntax und Semantik.

Das Anwendungsgebiete natürlichsprachlicher Schnittstellen sind nach Zoeppritz (1995):

- Abfragesprachen
- Kommandosprachen
- Maschinelle Übersetzung
- Sprache zu Text (oder Anfrage)
- Text (oder Antwort) zu Sprache
- Text Kritisierung (Hinweise zur Grammatik und zum Stil)
- Inhaltserfassung (von dem Filtern von Email nach dem Betreff bis zur Abstraktion)

Die Bedeutung der natürlichen Sprache als Kommandosprache ist aus meiner Sicht durch die zunehmende Verbreitung und Akzeptanz von grafischen Benutzeroberflächen in den letzten Jahren erheblich gesunken.

Shneidermann (1983) vergleicht die Möglichkeiten der grafischen interagierenden gegenüber den konventionellen (kommandoorientierten) Systeme mit dem Autofahren: Anstatt die Funktionstasten zu benutzen um die gewünschte Richtung („links“, „rechts“, Angabe des Winkels) zu bestimmen oder eine natürlichsprachliche Eingabe zu machen („drehe das Lenkrad um 30 Grad nach links“ ...), drehen wir das Lenkrad selbst. Wir erhalten so eine sofortige Rückmeldung über die Änderungen, die unsere Aktion ausgelöst hat, und können die benötigten Korrekturen ausführen.

Im weiteren werden von den genannten Anwendungsgebieten die Abfragesprachen und Sprache zu Text (oder Anfrage) im Vordergrund stehen, weil sie für das Thema „natürlichsprachliche dialoggesteuerte Datenbankabfrage“ am wichtigsten sind.

2.2.1 Vor- und Nachteile natürlicher Sprache

Es gibt viele Argumente die auf die Möglichkeiten der natürlichen Sprache an maschinellen Schnittstellen eingehen. Im Folgenden werden einige häufig genannte und einige im Rahmen dieser Arbeit interessante Argumente noch diskutiert. Viele der vermeintlichen Vorteile lassen sich genauso widerlegen wie die meisten Nachteile, so dass geprüft werden muss, ob die natürliche Sprache für die entsprechende Anwendung ein geeignetes Medium darstellt.

Man sollte keine Antwort auf die Frage erwarten, ob natürlichsprachliche Schnittstellen generell den formalen Sprachen überlegen sind. Auch neben grafischen Schnittstellen, die mit Methoden der direkten Manipulation arbeiten gibt es eine Nische in der die natürliche Sprache eine effektiveres Interface darstellt.

Im Folgenden werden einige der vermeintlichen Vor- und Nachteile vorgestellt:

Die Vorteile:

1. Es gibt eine 1:1 Analogie zwischen dem Verhalten an natürlichsprachlichen Schnittstellen und der Kommunikation zwischen Menschen (vgl. Grishman, 1986 und Kannengießer, 1989)
2. Die natürliche Sprache muss von den Benutzern nicht erlernt werden, da sie diese schon beherrschen. (Woods, 1972, Seite 522)
3. Ausdrücke die in formalen Sprachen wie SQL sehr komplex sind, lassen sich in der natürlichen Sprache einfach formulieren. (Zoeppritz, 1983)
4. Der Benutzer kann an NL-Schnittstellen sein Informationsbedürfnis so formulieren wie es ihm einfällt und ist nicht durch seine Kenntnis der formalen Sprache eingeschränkt. (Woods, 1972, Seite 522)
5. Natürlichsprachliche Schnittstellen können einen effektiven Zugang zu großen Datenmengen bieten.

Die Nachteile:

I. Natürliche Sprache ist wortreicher als formale Sprachen.(Woods, 1977)
II. Natürliche Sprache ist redundant.
III. Natürliche Sprache ist nicht so präzise wie formale Sprachen. Natürliche Sprache enthält Ambiguitäten, und Dialoge enthalten Fehler und ungrammatikalische Äußerungen. (Zoeppritz, 1983)
IV. Entgegen der unter 2. vorgebrachten Behauptung muss die natürliche Sprache an NL-Schnittstellen doch erlernt werden.
V. Die natürliche Sprache kann nicht umfassend implementiert werden. Es ist immer nur ein Teil der Sprache, der an einer Schnittstelle verfügbar ist. Es ist auch noch auf lange Zeit mit gravierenden syntaktischen und semantischen Einschränkungen zu rechnen. (Zoeppritz, 1983)
VI. Der Aufwand, eine natürlichsprachliche Schnittstelle zu programmieren, ist erheblich höher als bei formalen Sprachen.

VII. Wenn man dem Benutzer die Möglichkeit gibt, selber fehlende Vokabeln zu definieren, werden immer wieder weitere Wörter hinzukommen, und es wird dem Benutzer nicht gelingen, zu einer korrekten und abschließenden Definition kommen. (siehe Krause, 1988)

VIII. Die Methoden der direkten Manipulation bieten einfachere und effizientere Schnittstellen als die natürliche Sprache.

Einige der hier genannten Punkte schließen sich schon gegenseitig aus und die Frage welches der Argumente besser begründet ist, kann wohl auch nicht abschließend beantwortet werden.

So widerspricht 3. ganz klar I. und II. und für beide Seiten lassen sich genügend Beispiele anführen (siehe z.B. das Beispiel im Kapitel 2.1). Zoeppritz (1983) führt an, dass Vergleiche von natürlichsprachlichen Eingaben und daraus erzeugten ISBL[3]-Ausdrücken in etwa gleiche Längen ergeben haben. SQL-Ausdrücke sind im Vergleich zu ISBL-Ausdrücken noch länger.

Auch die Argumente 2. und IV. sind einander entgegengesetzt. An dieses Problem schließt sich das Argument V. an, dass ein NL-System nur eine Teilmenge der natürlichen Sprache kennt. So ist an dieser Stelle die Qualität der Implementation wichtig. Die Programmierer müssen alle sprachlichen Varianten erfassen. Tun sie dies nicht, ist es für den Benutzer unerlässlich, die Einschränkungen (möglicherweise vorher) zu erlernen. Im Q&A System, das im Kapitel 2.1 noch vorgestellt wird, sind, obwohl es sich um ein NL-System handelt, Phrasen vorgegeben, die der Benutzer erlernen muss, um die Funktionen nutzen zu können. Es gibt aber auch Einschränkungen, die der Benutzer während des Umgangs mit dem System erlernen kann. Es ist nicht zu leugnen, dass ein Benutzer den Umgang mit einem NL-System erlernen muss, wie den Umgang mit jedem anderen Programm auch. Nur dürfte der Lernaufwand erheblich niedriger sein, als bei einer formalen Sprache, und es ist ein Ziel der Implementation, dafür zu sorgen.

Dem Argument 1., der 1:1 Analogie, wiederspricht Krause (1995) mit seiner Computertalk-Theorie, die im Kapitel 2.5.3 noch vorgestellt wird. Kurz gesagt untersuchte Krause das Verhalten von Benutzern an NL-Schnittstellen und stellte fest, dass eine stark vereinfachte

3 Eine formale Sprache zur Datenbankabfrage wie SQL.

Sprache in der Kommunikation mit dem Computer verwendet wird, auch wenn dazu keine Notwendigkeit besteht.

Von einer 1:1 Analogie an NL-Schnittstellen ist aufgrund der Beschränkung auf eine Teilmenge der Sprache sowieso nicht auszugehen (siehe Argument V.)

Auch die Argumente 4. und V. können einander gegenübergestellt werden. Der Benutzer muss die Restriktionen, die für die verarbeitbare Sprache gelten, beachten, was es ihm vielleicht nicht ermöglicht, seine Anfrage so zu formulieren, wie er es gerne möchte. Das ist einer der Ansätze dieser Arbeit, da eine der verbreitetsten Einschränkungen die Beschränkung auf eine Äußerung als Eingabe ist. Vielen Benutzern dürfte es in der normalen Sprache fremd sein, mehrere Fragen in einer Anfrage zu kombinieren.

Zu V. sollte noch gesagt werden, dass Benutzer in der Lage sind, mit semantischen und syntaktischen[4] Einschränkungen umzugehen. Es dürfen bloß keine Lücken in der Implementierung vorhanden sein, so dass eine Funktionalität nur unter gewissen Umständen verfügbar ist. Wenn eine Funktion durchgängig nicht vorhanden ist, werden die Benutzer lernen, diese zu vermeiden. Dies gilt natürlich nicht für alle Funktionen, eine gewisses Basiswissen wird, je nach Applikation, erwartet. (Zoeppritz, 1983)

Dies zeigt die Möglichkeit auf, dass eine Einschränkung der Sprache erfolgreich sein kann, wenn man vernünftige Grenzen zieht.

Der höhere Aufwand für die Implementierung einer natürlichsprachlichen Schnittstelle (Argument VI.) schreckt sicherlich viele davon ab, für ein Projekt zu prüfen, ob ein solches Interface bessere Ergebnisse liefern könnte. Eine universelle portable NL-Schnittstelle, die den Aufwand erheblich verringern würde, ist nicht verfügbar, z.B. wegen der Beschränkung auf eine Teilmenge der natürlichen Sprache.

Den negativen Auswirkungen eines Benutzervokabulars (VII.) widerspricht Krause (1988) durch seine Studien am USL-System[5]. Der Umfang des zu definierenden Benutzervokabulars reichte von 49 bis 313 Worten. Die Benutzer waren durchaus in der Lage, Wortdefinitionen relativ vollständig und fehlerfrei vorzunehmen.

4 Wobei semantischen Einschränkungen weniger Probleme machen als Syntaktische.
5 Siehe Kapitel 3.1

Zuletzt lassen sich noch 5. und VIII. in Beziehung setzten. In vielen Bereichen, z.B. Kommandosprachen, habe grafische Interfaces die formalen Sprachen verdrängt und lassen aufgrund ihrer überlegenen Bedienbarkeit auch natürlichsprachliche Lösungen nicht erfolgversprechend erscheinen. Formale Sprachen werden bei den erwähnten Kommandosprachen noch von Experten genutzt, die mit dieser Sprache sehr effektiv umgehen können. Dies ist aber nicht die Zielgruppe eines NL-Systems, dessen Vorteile ja ehr in einem leichten Einstieg und geringem Lernaufwand liegen sollen.

Experten, die eine formale Sprache beherrschen, haben Probleme, sich auf eine natürlichsprachliche Eingabe umzustellen, da sie dazu neigen, Notationen aus der formalen Sprache zu übernehmen, auch wenn diese in der natürlichen Sprache nicht erlaubt sind. (vgl. Ronthaler, 2000)

Formale und natürliche Sprachen bieten aber große Vorteile, wenn es darum geht, mit großen Datenmengen umzugehen. Bei den Kommandosprachen kann man dies mit einem Beispiel verdeutlichen wie „Kopiere alle Dateien mit einem 'a' im Namen nach ...“, das mit Kommandosprachen, egal welcher Art, relativ leicht auszuführen ist, aber mit grafischen Oberflächen umständlich und manuell gelöst werden muss, da die Oberflächen nicht für solche Aufgaben optimiert sind.

Und auch Datenbanken, bieten große Datenmengen, welche sich in vielen Fällen nur schwer visualisieren lassen.

Um eine natürlichsprachliche Aussage umgehen zu können muss das System nicht in der Lage sein die gesamte Aussage verarbeitet zu können. Es ist möglich nur Teile des Aussage zu parsen und die Teile die nicht verarbeitet werden können zu vernachlässigen. In einem günstigen Fall enthalten die ausgelassenen Teile redundante Informationen oder die fehlenden Informationen lassen sich aus dem Kontext entnehmen, aber auch wenn dies nicht der Fall ist kann versucht werden mit den aus den verarbeitbaren Teilen erhaltenen Informationen weiterzuarbeiten. Dem Benutzer kann bei Rückfragen bzw. Fehlermeldungen z.B. mitgeteilt werden welche Informationen das System entnehmen konnte, so das nur der fehlende Teil noch einmal geäußert werden muss.

2.2.2 Analyse des Satzes

Ein Satz kann auf mehreren Ebenen analysiert werden. Zum einen auf der syntaktischen. Man legt eine Grammatik zugrunde und prüft, ob der Satz nach dieser auch korrekt ist.

Will man aus dem Satz aber auch Informationen zur Handlungssteuerung entnehmen, wird man auch die Bedeutung der Wörter und ihre Beziehung zueinander untersuchen müssen. Zu dieser semantischen Analyse werden in vielen Fällen auch die syntaktischen Informationen aus dem ersten Analyseschritt benötigt; das es anders geht, wird im Anschluss daran gezeigt.

Ein noch weitergehender Ansatz wäre eine pragmatische Analyse des Satzes. Dass dies momentan aber noch nicht machbar und es ist auch nicht klar ob es überhaupt möglich ist.

Um die syntaktische Struktur zu prüfen, gibt es verschienden Ansätze bis hin zu der Möglichkeit, auf eine syntaktische Analyse ganz zu verzichten und nur über die semantische Bedeutung der einzelnen Worte den Satz zu erschließen.

2.2.2.1 Syntaktische und semantische Analyse

Bei der syntaktischen Analyse wird der Satz in seinen Satzteile zerlegt. Eine syntaktische Analyse wird selten auf größeren Strukturen als einem Satz angewandt. Und deshalb wird ein Satz nach dem Anderen analysiert.

Ein Satz wird zuerst in Nominalphrasen (NP) und Verbalphrasen (VP) aufgeteilt. Die Nominalphrasen dann wieder in Artikel, Partikel, Nomen, Adjektive, Pronomen, usw. und die Verbalphrasen in Adverbien, Partikel, Verben, usw. unterteilt.

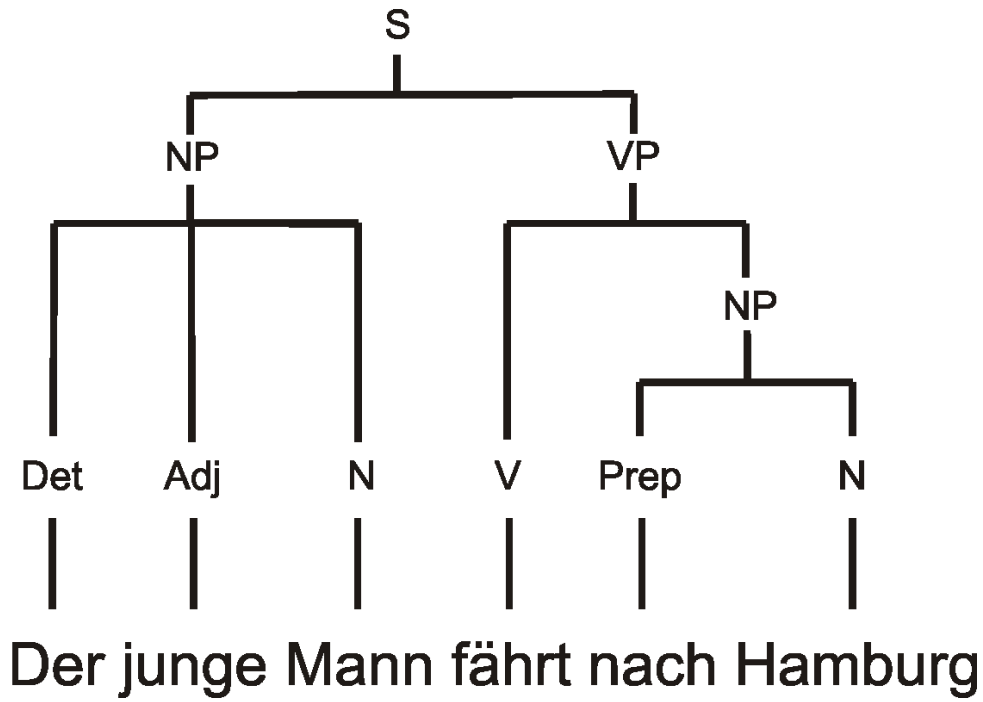


Abbildung 2.1 Struktur eines Satzes

Der in Abbildung 2.1 dargestellte Baum kann durch folgende Definit Clause Grammar (DCG) geprüft werden:

```
s --> np, vp.
vp --> v, np.
np --> det, adj, n.
np --> prep, n.
n --> [mann].
n --> [hamburg].
v --> [fährt].
adj --> [junge].
prep --> [nach].
```

Die Worte in eckigen Klammern sind lexikalisch Einträge. In dieser Form müssten alle Worte verfügbar sein, die der Parser kennt. Die hier angegebene Grammatik ist nur ein Beispiel; eine richtige Grammatik benötigt deutlich mehr Regeln. Um die syntaktische Korrektheit zu erfassen, werden noch andere Eigenschaften der Worte, wie z.B. Kasus, Numerus und Genus, benötigt. Diese können durch Erweiterungen der DCG berücksichtigt werden. Zum Beispiel:

```
np(Numerus) --> det(Kasus,Numerus,Genus),
                adj(Kasus,Numerus,Genus),
                n(Kasus,Numerus,Genus).
n(nominativ,singular,maskulinum) --> [mann].
```

Damit ein Programm auch weiß, was es mit der natürlichsprachlichen Eingabe machen soll, muss die Bedeutung der Aussage gefunden werden. Es reicht nicht, jedem Wort seine Wortart zuzuordnen und eine korrekte syntaktische Struktur des Satzes aufzubauen, die Semantik jedes Wortes muss bekannt sein. Die semantischen Informationen müssen lexikalisch verfügbar sein.

Verben können als Funktionen aufgefasst werden. „Geben“ zum Beispiel ist eine dreistellige Funktion. Wobei die Bedeutung von „geben“ der physikalische Transfer eines Objektes von einer Person an eine Andere ist.

```
X gibt Y an Z.
X ist die Person die die Handlung ausführt.
Y ist das Objekt, das transferiert wird.
Z ist der Empfänger.
```

Substantiven muss ebenfalls ihre Bedeutung zugewiesen werden. Sie können Eigennamen, Gegenstände, Ereignisse, abstrakte Konzepte (wie Frieden) und so weiter sein.

Adjektive dienen im allgemeinen dazu, Gegenständen oder Ereignissen Eigenschaften zuzuweisen. Die Bedeutung des Adjektives muss dem Wort zu dem es gehört, auch korrekt zugeordnet werden.

Manchmal hat eine Phrase eine weitergehende oder andere Bedeutung als die Wörter aus denen sie besteht (z.B. auf den Wecker gehen). Diese Bedeutung der Phrasen sollte natürlich auch im Lexikon gespeichert sein.

Die Relationen der Wörter müssen ausgewertet werden. Dazu bedarf es der Auswertungsergebnisse der syntaktischen und semantischen Analyse.

Wie genau die Bedeutung der Wort erfasst werden muss, hängt von der Aufgabe des Programmes ab. Ein Programm zum automatischen Übersetzen muss die Bedeutung genauer erfassen als ein Rechtschreibkorrekturprogramm.

Um die Bedeutung mancher Aussagen erfassen zu können, bedarf es Weltwissens. Das Problem ist, dass Weltwissen kaum oder gar nicht umfassend erfasst werden kann.

Wegen der Probleme mit dem Weltwissen ist eine pragmatische Auswertung fast nicht zu leisten. Aus Sätzen wie „Es zieht“ zu schließen, dass das Fenster oder die Tür geschlossen werden soll, ist für einen Rechner fast unmöglich.

In eine DCG lassen sich sehr gut Klauseln einfügen mit denen man unbekannte Namen, Straßen und Orte erkennen kann. Durch das beachten der Syntax und Semantik läßt sich oft recht genau erkennen um was es sich bei einem unbekannten Wort handelt. Deswegen habe ich mich auch bei dem Telefonauskunftsprogramm für dieses Verfahren entschieden.

Als Beispiel ein Ausschnitt aus einer möglichen DCG:

```
v --> [suchen], name.
name --> anrede, vorname, nachname.
name --> anrede, nachname.
name --> vorname, nachname.
name --> nachname.
anrede --> [herr].
anrede --> [frau].
vorname --> [VORNAME].
nachname --> [NACHNAME].
```

VORNAME und NACHNAME sind hier Variablen in denen die Werte für die Vor- und Nachnamen enthalten sind.

Leider wird durch das zulassen von unbekannten Worten in die DCG die Anzahl der möglichen Ergebnisse teilweise bis ins Unendliche gesteigert, so dass nicht mehr alle Alternativen berechnet werden können. Die DCG muss somit so programmiert werden, dass in

den ersten Alternativen möglichst die richtige Lösung enthalten ist, was leider nicht immer möglich ist.

2.2.2.2 ATN-Grammatiken

Augmented Transition Network Grammatiken können weitreichende Abhängigkeiten zwischen den Teilen eines Satz modellieren. Die zentrale Idee besteht darin, den Wörtern und Satzteilen Eigenschaften, wie Numerus, Genus oder die Person von Substantiven und Verben, zuzuordnen. Die möglichen Werte für diese Eigenschaftsvariable werden für jedes Wort in einem Lexikon gespeichert. (Potthast, 2001)

Man kann sich ATN als ein Netz vorstellen, das zwischen jedem Wort eines Satzes, mit dessen möglichen Wortarten und den daraus folgenden Eigenschaften, in Beziehungen zu den anderen Worten des Satzes aufgespannt ist. Zwischen diesen Worten gibt es Kanten, jede dieser Kanten steht für eine mögliche Belegung.

Über den Kanten im AT-Netz werden nun Tests durchgeführt, die erfolgreich sind wenn ein Wert oder eine Liste von Werten zurückgeliefert werden kann. Wenn ein Test nicht erfolgreich ist, wird die getestete Kante des Netzes und die nach ihr Folgenden nicht weiter verfolgt. (Potthast, 2001)

Der Formalismus wird noch um Tests erweitert, die z.B. prüfen, ob ein Verb in der Eigenschaft Person mit schon geprüften Substantiven übereinstimmt. Wenn die Schnittmenge der Verben und Substantive leer ist, müssen über Backtracking neue Ergebnisse gefunden werden. (Potthast, 2001)

Während der Traversierung des AT-Netzwerkes wird die linguistische Information mit prozeduralen Operationen (z.B. der Bearbeitung von Registern) vermischt (Varges, 1997, Kapitel 1). „Ein ATN ist eher eine Beschreibung von einem Prozesses zum Erkennen einer Sprache, denn ein Modell der Sprache selbst.“ (Pereira und Warren, 1980, S.120)

ATN-Grammatiken werden von Q&A-System, das in Kapitel 3.2 noch vorgestellt wird, eingesetzt. Für das Telefonauskunftsprogramm habe ich sie nicht in Betracht gezogen.

2.2.2.3 Mustererkennung

Die mit Abstand einfachste Möglichkeit zur Analyse eines eingegebenen Satzes ist die nach vorher definierten Schlüsselwörtern oder Phrasen im Satz zu suchen, die in dem

entsprechenden Kontext auftauchen können. Die erwarteten Muster müssen vorher bekannt und kodiert sein. Durch die Benutzung einer lexikalischen Komponente können natürlich auch Synonyme für die Muster leicht in die Suche miteinbezogen werden. Bei dieser Methode ist es auch ratsam eine, morphologische Komponente zu benutzen, damit nicht sämtliche Flexionen einen Wortes gespeichert sein müssen.

Wenn man sein Wissen intelligent strukturiert, kann man mit Suche nach Mustern recht gute Ergebnisse erzielen. So setzen Patrick und Whalen (1992) im COMODA-System, welches in Kapitel 3.5 vorgestellt wird, eine Hierarchie ein, in der an jeder Kante Muster und an jedem Knoten Dokumente gespeichert sind. Folgt man dieser Kante, kann am nächsten Knoten geprüft werden, ob im Satz noch weitere Muster gespeichert sind, die mit den, an den vom aktuellen Knoten ausgehenden, Kanten gespeicherten Mustern übereinstimmen. So kann man mit dieser Hierarchie ein Dokument finden, das möglichst optimal zu dem Satz passt.

Phrasen müssen nicht unbedingt geschlossen auftreten, sondern können auch so kodiert sein, dass die einzelnen Worte getrennt im Satz auftreten. Dieses Vorgehen ist z.B. von Wildcards (*) in der DOS-Shell bekannt.

Als eine Alternative kann eine Liste von Wörtern geführt werden, die ignoriert werden. Die restlichen Wörter werden als relevant weiterverarbeitet.

Namen in natürlichsprachlichen Sätzen zu erkennen ist mit der Mustererkennung nicht immer möglich. Es ist möglich eine Liste mit Vor- und Nachnamen zu führen und diese Muster in der Eingabe zu suchen, aber vor allem bei Nachnamen (z.B. Klein, Baum) kann es auch schnell zu Mehrdeutigkeiten kommen. Auch wenn mehrere Namen in einer Aussage vorkommen gibt es Probleme, so dass z.B. nicht die Beziehung dieser beiden Namen zueinander aufgelöst werden kann. Unter anderem aus diesen Gründen ist diese Methode für das Telefonauskunftssystem nicht anwendbar, da dort die Erkennung von Eigennamen, Orts- und Straßennamen von entscheidender Bedeutung ist.

2.2.3 Linguistische Phänomene

Einen guten Überblick über linguistischen Phänomene findet man bei Ronthaler (2000). Hier wird im Folgenden nur auf einige Fälle eingegangen. Die Auswahl erfolgt zum einen aufgrund der Häufigkeit des Phänomens in der Sprache und zum anderen wurden Phänomene ausgewählt, die im Rahmen dieser Arbeit eine Rolle spielen.

2.2.3.1 Anaphorische Referenz[6]

In der Sprache gibt es typischerweise Pronomen oder definite Nominalphrasen die als Referenz auf einen anderen Ausdruck verweisen. Dies ist normalerweise ein vorangegangener indefiniter Ausdruck. (Ronthaler, 2000)

```
Ein Haus steht auf dem Hügel. (Es/Das Haus) wird bald abgerissen.
```

„Es“ und „das Haus“ beziehen sich auf den selben Referenten, sie koreferieren. „Ein Haus“ heißt Antezedent, weil es dem Pronomen „es“ vorangeht. (Ronthaler, 2000)

Anaphern werden nach Hirst (1981) nur eingesetzt, wenn nach Ansicht des Sprechers der Hörer in der Lage ist, die Referenz auch aufzulösen.

Hirst gibt einige Möglichkeiten an, wie das Antezedent gefunden werden kann:

- Das Antezedens wird explizit im Text erwähnt und ist „nicht allzuweit“ zurückliegend.
- Das Antezedens ist im vorangehenden Text implizit enthalten und wieder „nicht allzuweit“ zurückliegend.
- Das Antezedens wird mit einer Geste oder anderen außersprachlichen Methode eingeführt oder ist nach Ansicht des Sprechers in der jeweiligen Situation bekannt.

Die Auflösung einer Anapher ist aber nicht immer möglich. So kann es z.B. zu Ambiguitäten kommen, weil sich der Antezedent nicht eindeutig bestimmen lässt. Bei Computern spielt es eine Rolle, dass diese über kein umfassendes Weltwissen verfügen und es so auch nicht immer möglich ist eine Referenz aufzulösen. Die Verarbeitung der außersprachlichen Referenzen ist für Computer normalerweise nicht möglich.

Deutlich seltener als die anaphorische Referenz ist die kathaphorische Referenz in der ein Pronomen oder Adverb dem Teil vorausgeht, auf den Bezug genomen wird. (Duden, Bd.4, Kapitel 571,637) Beispiel: Ich suche *einen* in Hamburg, *der Peter Meier heißt*.

In Dialogen tauchen anaphorische Referenzen häufig auf wie folgendes Beispiel zeigt:

A: „Ich suche *Peter Meier*.“
B: „In welchem Ort wohnt *er*?“
A: „*Der* wohnt in Berlin.“

Im initialen Zug wird eine Person eingeführt auf die beide Sprecher in den folgenden Zügen durch anaphorische Referenzen verweisen.

6 Vergleiche Ronthaler, 2000, Kapitel 4.2.2

2.2.3.2 Ellipsen

Ellipsen lassen sich als nicht in der Oberflächenstruktur eines Satzes vorhandene Elemente definieren. Aber nicht jedes nicht realisierte Element eines Satzes darf als Ellipse angesehen werden. (Ronthaler, 2000, Kapitel 4.2.3)

Es handelt sich bei einer Ellipse um Auslassungen in einer Äußerung, die vom Hörer aus dem Kontext rekonstruiert werden können. (Block und Schachtl, 1995)

Ronthaler gibt nach Thomas (1979) drei Klassen von Auslassungen an, von denen aber nur eine wirklich eine Ellipse ist:

- Elision: Hier wird weggelassen, was durch das sprachliche Wissen des Hörers erschlossen werden kann.
- Ellipse: Die Informationen, die ausgelassen wurden, können durch den Kontext erschlossen werden.
- „Nonrealization“: Unterscheidet sich von den anderen beiden dadurch, dass weder durch sprachliche Konvention noch durch den Kontext auf das fehlende Element geschlossen werden kann.

Im weiteren weist Ronthaler darauf hin, dass Äußerungen durch Ellipsen ambig werden können. Dazu bringt er das Beispiel „Peter betrachtet sich im Spiegel. Maria auch“. Die zweite Aussage läßt nicht erkennen, ob Maria sich selbst oder Peter im Spiegel betrachtet oder von Peter im Spiegel betrachtet wird.

Block und Schachtl (1995) trennen drei Arten der Ellipse: Topic Ellipsis, Adjacency Ellipsis, und Reduktion auf das Normale. Reduktion auf das Normale führen sie nicht weiter aus, es dürfte sich aber um das selbe Phänomen wie die Elision handeln.

- Topic Ellipsis ist zum einen das Weglassen von Füllworten und Ähnlichem am Satzanfang und zum anderen das Auslassen von Pronomen, die entweder Subjekt, Akkusativ oder Dativ Objekt bzw. ein substituierendes „das“ sind.
- Adjacency Ellipsis tritt im besonderen bei Fragen auf. Redundante Informationen werden in der Antwort nicht wiederholt. Die Frage und die Antwort können als ein zusammenhängendes Paar gesehen werden, in dessen zweiten Teil die redundante Information ausgespart wird.

Block und Schachtl beschreiben auch noch das Problem der „Selbständigen Setzung“, bei der Hauptsätze durch Phrasen ersetzt werden, die kein bestimmtes Verb enthalten. Die fehlenden Worte können auch nicht aus dem Kontext erschlossen werden. Das Phänomen

ähnelt der Elision von Thomas und wird auch von Block und Schachtl nicht direkt zu den Ellipsen gezählt.

Ellipsen sind in Dialogen sind häufig. Das Beispiel aus dem letzten Kapitel würde wie folgt aussehen, wenn Ellipsen und Elisionen dabei eingesetzt würden:

A: „Ich suche Peter Meier."
B: „In welchem Ort?"
A: „Berlin."

Das Auslassen von „wohnt er" in der Frage von B ist eine Elision, da diese Information durch den Kontext und die Semantik von Ort erschlossen werden kann. Das Fehlen von „Der wohnt in Berlin ist eine Adjazenz Ellipse, da redundante Informationen ausgespart wurden (aus wenn in diesem Fall diese Informationen nicht einmal in der Frage auftauchten).

2.2.3.3 Sprechakte

Bei Sprechakten geht es darum, wie man etwas mit Worten macht. Austin (1962) unterscheidet den lokutionären Akt, den illokutionären Akt und den perlokutionären Akt.

Wenn einfach etwas gesagt wird, handelt es sich um einen lokutionären Akt. Eine Äußerung, die in der Absicht geäußert wird, damit eine Handlung auszuführen, ist ein illokutionärer Akt, und ein perlokutionärer Akt ist eine Äußerung bei der eine Wirkung auf den Hörer erwartet wird.

Eine Äußerung kann nach Austin (1962) auch noch weiter in eine phonetische, phatetische und rhetische Ebene unterteilt werden. Die phonetische Ebene ist das Äußern von Lauten, die phatetische das Äußern von Worten und Sätzen und die rhetische die Äußerung der Bedeutungen.

Es gibt indirekte Sprechakte, wie z.B. „Es zieht!", die eine andere Illokution haben, als die aus der Äußerung zu entnehmende, in diesem Fall statt einer Feststellung eine Aufforderung, das Fenster oder die Tür zu schließen. Eine Auflösung solcher indirekten Sprechakte ist für Maschinen, wie auch schon an diesem Beispiel dargestellt, nicht immer möglich.

Strobel (1997, Kapitel 2.3) weißt darauf hin, dass in neuer Zeit eine Sprechaktsequenztheorie entwickelt wird, die allerdings nur die Folge einiger weniger Sprechakte umfasst. Nach Strobel wird eigentlich eine Theorie der Sprech- und Hörerverstehensakte, die die Teile eines Kommunikationsaktes darstellen, benötigt. In den

Hörerverstehensakt muss laut Strobel auch noch die Antizipation des Sprechaktes durch den Hörer mit einbezogen werden.

In der später vorgestellten Implementation wird z.B. der Sprechakt „Tschüs“ als Verabschiedung des Benutzers interpretiert. Das System folgert daraus, dass der Benutzer nun mit seiner Arbeit fertig ist und deshalb wird das Programm automatisch beendet, nachdem eine Verbschiedung des Systems ausgegeben wurde.

2.2.3.4 Kontext

Aus schon im Gespräch getätigten Äußerungen können Ambiguitäten aufgelöst und, wie bei den Ellipsen erwähnt, fehlende Informationen ergänzt werden. Daneben können im Kontext handlungssteuernde Informationen enthalten gewesen sein. So bringt Ronthaler (2000) das Beispiel, dass ein Benutzer, der bei der Büchersuche einmal mitgeteilt hat, dass er nur deutschsprachige Literatur wünscht, bei seinen folgenden Anfragen auch nur deutschsprachige Literatur ausgegeben wird. Der Benutzer soll aber bei der Präsentation der Suchergebnisse auch darauf hingewiesen werden, dass diese Einschränkung noch immer gilt und ihm die Möglichkeit eingeräumt werden diese auch wieder zu ändern.

In dem Telefonauskunftssystem, das in Kapitel 4 vorgestellt wird, wird dies auch versucht, indem Informationen aus dem bisherigen Kontext in die vom System generierten Fragen und Antworten mit eingebaut werden.

Besonders bei Informationen, die nur indirekt aus dem Kontext gewonnen wurden, ist es meines Erachtens wichtig, diese Schlüsse dem Benutzer zu nennen.

Beispiel:

```
A: „Ich suche Peter Meier aus Berlin.“
B: „Da gibt es einen Herrn Peter Meier, Hauptstraße 12, ...“
A: „Und nun suche ich noch Klaus Schulz.“
B: „Wenn Sie Klaus Schulz aus Berlin meinen, dann wohnt der ...“
```

2.2.3.5 Unterschiedliche Bedeutungen von „Und“[7]

Da es in dieser Arbeit um die Datenbankabfrage geht, ist der Unterschied in der Verwendung der umgangssprachlichen „und“ gegenüber dem logischen „und“, das auch in strukturierten Anfragesprachen verwendet wird, natürlich von besonderem Interesse.

7 Vergleiche Ronthaler, 2000, Kapitel 4.2.14

So wird die Konjunktion „und“ umgangssprachlich, aber auch an Mensch-Maschine-Schnittstellen häufig mit disjunktiver Bedeutung verwendet (Das-Gupta, 1987; Ogden und Kaplan, 1986).

„Ich esse gerne Joghurt und Früchte“ ist eine mehrdeutige Aussage. Es kann sowohl Joghurt mit Früchten bedeuten, als auch, dass man Joghurt sowie die Früchte gerne einzeln isst.

An einer simulierten natürlichsprachlichen Schnittstelle stellten Ogden und Kaplan (1986) fest, dass die Benutzer das „and“ nicht gemäß den logischen Wahrheitswerten verwenden können. Das „and“ wurde in 30% der Fälle benutzt um die Vereinigungsmenge zu bilden und in über 99% wurde es in Sätzen benutzt in denen die Schnittmenge gebildet werden sollte.

Wenn also ein „and“ in einer natürlichsprachlichen Aussage auftritt ist zunächst einmal nicht klar, wie dieses zu interpretieren ist, da es die Wahrscheinlichkeit, dass für ein logisches „or“ ein natürlichsprachliches „and“ verwendet wurde sehr hoch ist.

Das natürlichsprachliche „or“ kann sicher als das logische „or“ interpretiert werden, beim natürlichsprachlichen „and“ muss man auf zusätzliche Informationen zurückgreifen, heißt es bei Ogden und Kaplan (1986). Weiterhin nennen sie Fälle in denen ein „and“ doch als logisches „and“ interpretiert werden kann:

1. „And which“ wurde benutzt („The donkeys living in an stable and which came from Ireland“).
2. Die Singularform des fehlenden Elementes wurde benutzt. („The student majoring in history and math“).
3. „Both“ wurde benutzt („The students majoring in both math and history“).
4. Ein Relativpronomen geht dem „and“ im Satz voraus („The students that major in math and history“)

Für das Deutsche können ähnliche Heuristiken gefunden werden.

Zum „Oder“ ist leider noch anzumerken, dass es auch das exklusive „Oder“ gibt, bei der nur eine der mehreren Alternativen wahr sein kann, z.B. „Du kannst den Kuchen essen oder behalten“. Für viele Benutzer ist dies die normale Anwendung des „Oder“, wohingegen das logische „Oder“ es auch erlaubt, dass alle Alternativen wahr sind.

Der Dialogansatz erlaubt es Formulierungen mit „und“ teilweise zu vermeiden, da das System dann die Relation zwischen den einzelnen Aussagen des Benutzers auflösen muss. Beispiel:

A: „Ich suche Peter Meier.“
B: „In welchem Ort wohnt Peter Meier?“
A: „In Berlin.“
B: „In welcher Straße wohnt Peter Meier?“
A: „Waldweg oder Waldstraße.“

Dieser Dialog entspricht folgendem Ausdruck der mit logischen Operatoren verknüpft ist:

Vorname(Peter) AND Nachname(Meier) AND Ort(Berlin) AND (Straße(Waldweg) OR Straße(Waldstraße)).

Die wahrscheinlichere Interpretation der letzten Dialogzeile ist, dass die gesuchte Person nur an einer der angegebenen Straßen wohnt und nicht Wohnungen an Beiden hat, so dass statt des OR auch ein XOR möglich wäre.

2.2.3.6 Weitere Phänomene

Der Vollständigkeit wegen werden hier noch einige weitere linguistische Phänomene kurz vorgestellt.

Ambiguität spielt an maschinellen Schnittstellen eine große Rolle. Da einem Computer kaum Weltwissen zur Verfügung steht, ist es nicht unter Umständen nicht möglich, Ambiguitäten aufzulösen, die einem Menschen noch nicht einmal auffallen. (Ronthaler, 2000, Kapitel 4.2.13)

Beispiel: „Klaus sucht Peter. Er wohnt in Berlin.“

Komposita sind zwei oder mehr Worte, die in der deutschen Sprachen zu einem Wort zusammengefasst wurden. Beim Trennen der Komposita muss das Wort richtig in seine Teile zerlegt werden und die korrekten Relationen zwischen den Teilen gefunden werden (Ronthaler, 2000, Kapitel 4.2.16). Auch im Englischen können Worte bei gemeinsamen auftreten eine andere Bedeutung haben als die einzelnen Worte für sich genommen. Im Gegensatz zum Deutschen werden sie zwar nicht zu einem Wort kombiniert, aber die Relationen zwischen den Worten aufzulösen ist genauso aufwendig.

Beispiel: „Ich suche ein Elektrofachgeschäft.“ – „Ich suche ein Fachgeschäft für Elektronik“

Präsuppositionen sind Bedingungen, die vor einer Aussage erfüllt sein müssen, damit diese Aussage wahr sein kann. Ein Beispiel: die Präsupposition von „beginnen“ ist, dass man die Tätigkeit mit der man gerade beginnt oder das Ereignis das gerade beginnt, vor diesem Zeitpunkt noch nicht ausgeführt wurden. (vgl. Ronthaler, 2000, Kapitel 4.2.4)

Deiktische Ausdrücke sind Referenzen auf situative Parameter. Es wird auf außersprachliche Objekte verwiesen, die in der Situation, in welcher die Äußerung gemacht wird, verfügbar sind. Dieses können Referenzen in der Zeit (z.B. „nächste Woche“) oder im Raum (z.B. „dort drüben“) sein. Um Deiktische Ausdrücke auflösen zu können, bedarf es einer programminternen Repräsentation des Raumes und der Zeit. (Ronthaler, 2000, Kapitel 4.2.8)

Beispiel: „Ich suche die Telefonnummer von Peter Meier. Er wohnt hier in der Straße.“

Conversational Filler sind Füllwörter, wie „hm“, „ah“, „ja“, „ok“, die nur in der gesprochenen Sprache auftauchen. Sie sollen auf Seiten des Sprechers darauf hinweisen, das ein Redebeitrag noch nicht beendet ist und auf Seiten des Hörers sind sie eine Reaktion, wie Bestätigung oder Zurückweisung. (Ronthaler, 2000, Kapitel 4.2.12)

Leider spielt bei Conversational Fillern oft auch die Intonation eine Rolle, wie das folgende Beispiel zeigt:

A: „ja,ähm, sie suchen Herrn Meier aus Osnabrück...“
B: „Hm!“ (zustimmend)
A: „... der in der Hauptstraße wohnt...“
B: „Hmhm!“ (ablehnend)
A: „Dann am Waldweg?“

Die Beiträge von B können gemacht werden während A noch redet, keine Reaktion würde als Zustimmung gewertet (siehe Kapitel 2.5.2) oder B könnte abwarten bis A seine Aussage beendet hat und dann Widerspruch einlegen.

Parenthese ist der syntaktisch vom Satz unabhängige Einschub in einen Satz. Die Einschübe können von von kleinen Bemerkungen bis zu ganzen Sätzen gehen. Parenthese tritt vorwiegend in der gesprochenen Sprache auf. (Block und Schachtl, 1995)

Beipspiel: „Ich suche – falls sie mir da helfen können – ein gutes griechisches Restaurant“

2.2.4 Kontextrepräsentation

Um Referenzen und Ellipsen erfolgreich auflösen zu können, ist es wichtig, den gesamten Kontext eines Dialoges zu erfassen und zu verarbeiten. Eine Beschränkung auf den Kontext eines Satzes ist nicht ausreichend.

Das im Rahmen dieser Arbeit entstandene Programm „Telefonauskunft“ benutzt lediglich die Semantik der schon gestellten Fragen als Kontext. Insbesondere die Bedeutung der letzten Frage spielt bei der Auswertung der Benutzereingabe eine Rolle. Die Auflösung des elliptischen Phänomens der Redundanzvermeidung bei der Beantwortung von Fragen ist auf diese Weise meistens möglich, andere Formen von Ellipsen und anaphorische Referenzen sind so nicht lösbar.

Als Beispiel einer umfassenderen Möglichkeit, den Kontext zu repräsentieren, wird im folgenden die Diskursrepräsentationstheorie (DRT) von Kamp (Kamp,1981; Kamp und Reyle, 1990) vorgestellt.

2.2.4.1 Diskursrepräsentationstheorie[8]

Die DRT erstellt anhand eines syntaktisch ausgewerteten Satzes eine Diskursrepräsentationsstruktur (DRS). Eine DRS K besteht aus einem Paar der Form <U(K),Con(K)>. Hierbei ist U(K) das Universum, d.h. die Menge der Diskursreferenten in der DRS K. Con(K) ist die Menge der Bedingungen über den Diskursreferenten in K. Wenn es für eine endliche Menge an Diskursreferenten $x_1,...,x_n$ ein n-stelliges Prädikat R gibt, dann ist $R(x_1,...,x_n)$ eine einfache Bedingung. (Ronthaler, 2000)

Durch einfache Bedingungen werden Substantive, Verben oder andere lexikalische Elemente des Satzes repräsentiert. Um Konditionalsätze, quantifizierte Ausdrücke, Negationen und Disjunktionen zu erfassen, werden komplexe Bedingungen in Form von Sub-DRSen benötigt. (Ronthaler, 2000)

In der DRT können bestehende Strukturen um neu hinzukommende Informationen jederzeit erweitert werden. Am Ende eines Satzes kann der bisherige Verlauf des Diskurses auswertet werden. (Ronthaler, 2000)

Ein Beispiele für die Anwendung der DRT:

„Ich suche Peter Meier. Er wohnt in Berlin“

Wertet man nun den ersten Satz aus, erhält man folgende Reprasentation:

8 Vergleiche Ronthaler (2000), Kapitel 4.2.1

x,y
Ich(x), Peter Meier(y), suche(x,y)

Wenn nun der zweite Satz zur DRS hinzugefügt wird kommen zwei neue Diskursreferenten einer für das Personalpronomen und einer für Berlin hinzu:

x,y,q,r
Ich(x), Peter Meier(y), suche(x,y) q = ?, Berlin(r), wohnt(q,r)

Im folgenden muss noch die anaphorischen Referenz aufgelöst werden.

x,y,q,r
Ich(x), Peter Meier(y), suche(x,y) q = y, Berlin(r), wohnt(q,r)

In diesem Beispiel ist es problemlos möglich die Referenz aufzulösen, wenn man aber das Beispiel nur leicht ändert kann das Personalpronomen nicht mehr eindeutig zugeordnet werden:

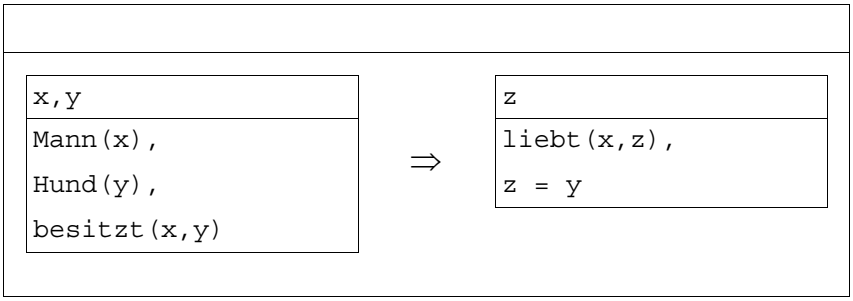
„Klaus sucht Peter Meier. Er wohnt in Berlin“

Mit „er“ kann sowohl Klaus als auch Peter Meier gemeint sein.

Um auch komplizierte Strukturen zu zeigen noch ein weiterer Beispielsatz:

„Jeder Mann der einen Hund besitzt, liebt ihn.“

Die Repräsentation dieses Satzes läßt sich so darstellen:



Die Sichtbarkeit der Diskursreferenten ist auf den Ort ihreres Auftretens beschränkt. Aus tieferliegenden DRSen kann auf die Referenten der höheren zugegriffen werden, aber nicht umgekehrt. Diese Einschränkungen sollen die Auflösung von anaphorischen Referenzen erleichtern.

Bei Dialogen muss beachtet werden, das durch den Wechsel des Sprechers einige Pronomen sich in der Person ändern, z.B. wird aus einem „Du“ ein „Ich“:

```
A: „Ich suche Peter Meier.“
B: „Können Sie mir sagen in welchem Ort er wohnt?“
A: „Er wohnt in Berlin.“
```

2.2.5 Phonetische Namenssuche

Normale Worte sind mehr oder weniger in ihrer Schreibweise standardisiert. In Lexika ist die korrekte Orthographie von Worten erfasst. Durch die Konventionen der geschriebenen Sprache wird man gezwungen sich möglichst an diese vorgegebene Schreibweise zu halten. Bei Personennamen ist die Zuordnung zu einer eindeutigen Schreibweise deutlich schwieriger. Dieselbe phonetische Repräsentation eines Namens kann verschiedenen Zeichenketten zugeordnet werden, z. B. „Maier“, „Meyer“. Die Festlegung, welche Zeichenkette nun die gesuchte ist unterliegt keinen allgemeingültigen Regeln, auch wenn manche Schreibweisen regional häufiger auftreten als andere.

Durch diese unterschiedlichen Repäsentationen gestaltet sich die Suche nach Personennamen oftmals schwierig (Hitzenberger, 1986).

Um eine einheitlichere Repräsentation von Namen zu erhalten, wurde Anfang des 20. Jahrhunderts der Russel-Soundex-Code entwickelt. Mit diesem Algorithmus war es möglich, gleich oder ähnlich klingende Namen wie „Smith“ und „Smythe“ auf den selben Soundex-Code (S530) abzubilden.

Soundex ist recht verbreitet und wird von vielen Datenbanken und Programmen angeboten und unterstützt. Leider ist der Soundex für amerikanische Namen optimiert und berücksichtigt auch nicht den Kontext in dem Namen auftreten. Deswegen wurde die „Kölner Phonetik“ (Postel, 1969) als angepasste Lösung für deutsche Namen entwickelt.

2.3 Generieren von Antworten und Fragen

Für eine adäquate Antwort auf eine Anfrage muss das System wissen, welche Informationen zur Beantwortung der Frage von Bedeutung sind. Auch der Computer sollte versuchen, den

Griceschen Maximen (Grice,1989) zu genügen. Deswegen sollen an dieser Stelle die Griceschen Maximen einmal vorgestellt werden.

Die Maxime der Quantität:

1. Mache deinen Beitrag so informativ wie in der momentanen Situation benötigt.
2. Mache deinen Beitrag nicht informativer als es benötigt wird.

Die Maxime der Qualität:

1. Sage nichts, von dem du glaubst, dass es falsch ist.
2. Sage nichts, von dem du nicht gute Gründe hast es zu glauben.

Die Maxime der Relation:

Sei relevant.

Die Maxime der Art und Weise:

1. Vermeide Unklarheiten
2. Vermeide Mehrdeutigkeiten
3. Fasse dich kurz
4. Schweife nicht ab

Vom Computer generierte Texte sollten sich an alle diese Maximen halten, auch wenn dies nicht immer möglich sein wird, da das System z.B. die Intention des Benutzers nicht richtig erfasst hat (Maxime der Relation) oder die Daten die ausgegeben werden müssen zu umfangreich sind (Maxime der Art und Weise und auch Maxime der Quantität). Das Einhalten der Maxime der Qualität sollte für einen Computer am einfachsten sein, da er alles was in seiner Datenbank steht für ihn Fakten und Glauben keine Rolle für ihn spielt.

Krause (1988) stellt fest, dass die Festlegung des Umfangs einer Antwort ein ernsthaftes Problem für jedes natürlichsprachliche Frage-Antwort-System ist. Er schlägt als Lösungsansatz eine Benutzermodellierung vor, welche vom Benutzer in einer bestimmten Situation nachgefragte Informationen in einer ähnlichen Situation automatisch ausgibt. Aber Krause sieht noch Forschungsbedarf in diese Richtung und hält es momentan (1988) noch nicht für realisierbar.

Aus meiner Sicht ist im Bereich der Benutzermodellierung seitdem einiges an Forschung geleistet worden, aber mir ist keine Arbeit bekannt, die sich mit genau diesem Thema beschäftigt. Aber wenn man z.B. einen geeigneten Korpus als Trainingsmenge hat, denke ich, dass es mit Techniken wie Self-Organizing Maps unter Umständen möglich ist, die durchschnittlich gewünschten Informationen auf eine Frage zu ermitteln.

Das Q&A-System (siehe Kapitel 3.2) löst dieses Problem durch globale Variablen, die vom Benutzer verändert werden können als auch durch vordefinierte Phrasen, die der Benutzer in seine Anfrage einarbeiten muss, um die Ausgabe zu verändern.

Wenn man ein natürlichsprachliches System benutzt, ist es natürlich auch zu bedenken, ob der Benutzer nicht auch eine natürlichsprachliche Antwort erwartet. Durch die Ausgabe von Tabellen oder ähnlichen Informationsstrukturen, wie es z.B. in Q&A und USL der Fall ist, wird der Benutzer eindeutig darauf hingewiesen, dass er mit einer Maschine kommuniziert.

Aber auch bei einer Maschine passt sich der Mensch an seinen Gesprächspartner an. So berichtet Krause (1988), dass selbst die Reihenfolge der Spalten in einer Tabelle vom Benutzer in einer anschließenden Anfrage berücksichtigt werden kann, selbst wenn der Satz dadurch grammatikalisch nicht mehr korrekt ist.

Wenn das Medium die Möglichkeit zur graphischen oder tabellarischen Darstellung bietet, kann es von großem Vorteil sein, Informationen zu visualisieren statt sie in Sätzen formulieren zu müssen. Vor allem bei größeren Informationsmengen wird die Verständlichkeit deutlich erhöht und Redundanz drastisch verringert. Auch die Geschwindigkeit, in der die Informationen vom Benutzer erfasst werden können, ist deutlich höher. Vom Programmieren her ist es deutlich weniger Aufwand, z.B. eine Tabelle auszugeben als aus den Informationen eine natürlichsprachliche Antwort zu generieren. So ist es bei Systemen mit der Möglichkeit zur Visualisierung vor allem abhängig von der Aufgabe des Programmes und der Menge der Informationen, die ausgegeben werden müssen, wie die Antwort gestaltet wird.

Falls eine Tabelle doch einmal verbalisiert werden muss sollte das System Strategien besitzen, wie dies effizient geschehen kann. Es ist nicht notwendig eine Tabelle Eintrag für Eintrag vorzulesen, stattdessen muss erkannt welches „Muster“ in der Tabelle enthalten ist (z.B. 90% der Einträge sind identisch) und welche Einträge für den Benutzer von besonderen interesse dabei sind weil sie von diesem Muster abweichen. Auch komplexere „Muster“, für die es einfache Formulierungen gibt, könnten erkannt werden. Nicht jeder Tabelleneintrag muss auch für den Benutzer interessant sein, ganze Spalten brauchen so möglicherweise nicht ausgegeben werden.

Falls die Kommunikation über gesproche Sprache, z.B. am Telefon, erfolgt, ist eine Visualisierung ausgeschlossen. Bei manchen Strukturen gibt es in der gesprochenen Sprache Konventionen, wie diese Aussagen verkürzt werden können, so ist es z.B. nicht nötig, eine

Anschrift als „Ich habe hier einen Herrn mit dem Vornamen Karl und dem Nachnamen Meyer, der in der Hauptstraße Nummer 17 wohnt. Die Postleitzahl ist 12345 und der Ort heißt Neustadt.“ anzugeben und „Herr Karl Meyer (Pause) Hauptstraße 17 (Pause) 12345 Neustadt“ ist wohl genauso verständlich. Aber auch Strukturen, die nicht so häufig in der gesprochenen Sprache vorkommen, müssen in Sätzen formuliert werden[9].

Bei Ausgaben in gesprochener Sprache ist die Länge der Fragen und die Menge der darin enthalten Informationen bzw. Auswahlmöglichkeiten zu bedenken. Wenn z.B. zu viele Auswahlmöglichkeiten in einer Frage genannt werden, kann es sein, dass der Benutzer den Überblick verliert, weil er sich nicht alle Optionen merken konnte. Strobel (1997, Seite 105) sagt, dass bis zu 5 Alternativen für den Benutzer handhabbar sind, wenn er in natürlicher Sprache antworten kann. Falls den Alternativen Zahlen zugeordnet werden, sind nur 3 Menüpunkte zu empfehlen.

Strobel (1997, Seite 15) hält eine natürlich klingende Stimme an maschinellen Schnittstellen für kontraproduktiv, da die von den Benutzer verwendeten Einschränkungen der Sprache an diesen Schnittstellen, vom System durchaus gewünscht sein können. Alleine der von Strobel beschriebene Verzicht auf Höflichkeitsfloskeln reduziert den Verarbeitungsaufwand. Auf die unterschiedliche Sprache an Mensch-Maschine-Schnittstellen wird in Kapitel 2.5 noch eingegangen.

In einem Dialogsystem sind vom Computer aber nicht nur Antworten zu formulieren, sondern auch Nachfragen. Das Spektrum der Möglichkeiten geht von „Straße?“ bis hin zu „Können Sie mir sagen in welcher Straße Herr Meyer wohnt?“. Wobei für die Fragen meines Erachtens indirekte Sprechakte wie in der zweiten Frage vermieden werden sollten, da sie unter Umständen einen weiteren unnötigen Dialogschritt bedeuten. Informationen aus dem Kontext in die Frage mit aufzunehmen hat mehrere Vorteile, zum einen kann dem Benutzer noch einmal der Kontext, in dem diese Frage steht, verdeutlicht werden und zum anderen kann dem Benutzer die Möglichkeit gegeben werden diese Information zu korrigieren, falls sie nicht mehr zutreffend ist (vgl. Ronthaler, 2000, Kapitel 4.2.9). Ein Beispiel hierfür wäre:

```
A: „Ich suche Peter Meier.“
B: „In welchem Ort wohnt Peter Meier?“
A: „Moment, ich glaube der heißt Mayer und nicht Meier.“
```

9 Beispiel einer Computerhotline. Der Ausdruck „Da müssen sie Start drücken, dann Systemsteuerung wählen und in dem Fenster dann Netzwerk auswählen. Und nun klicken sie mit der rechten Maustaste TCP-IP an und wählen Einstellungen.“ entspricht ungefähr „Da müssen wir die TCP-IP-Einstellungen ändern“, was aber nur Experten sofort in die oben genannte Liste von Aktionen umsetzten können.

Auch wenn Nachfragen keine Informationen enthalten müssen, werden sie doch aufgrund von Fakten ausgewählt. So ist es von Bedeutung, welche Informationen schon abgefragt wurden, welche der noch nicht gestellten Fragen überhaupt einen Informationsgehalt haben und welche Informationen der Benutzer überhaupt anbieten kann[10].

Aufgrund der Anpassung des Benutzers (vgl. Krause, 1988, Zoeppritz, 1983) an die Formulierungen des Gesprächspartners ist es nötig, die Fragen so zu formulieren, dass die zu erwartenden Antworten auch vom System verarbeitet werden können.

Um Antworten effizienter weiterverarbeiten zu können, ist es gelegentlich ratsam, Fragen so zu formulieren, das minimale Antworten zu erwarten sind.

„Die Minimalantwort stellt den Gegenzug dar, der von einem Dialogsystem als derjenige mit der größten 'Zielgenauigkeit' und der geringsten Redundanz am stärksten präferiert wird." (Strobel, 1997, S. 64)

Eine Minimalantwort kann z.B. erzeugt werden indem die Fragestellung Adjazenz Ellipsen provoziert.

Auswahl- oder Entscheidungsfragen neigen ebenfalls dazu Minimalantworten zu erzeugen, wenn man sie richtig formuliert. Auch an dieser Stelle kann das Phänomen ausgenutzt werden, dass Formulierungen vom Dialogpartner übernommen werden.

Beispiele für Fragen die Minimalantworten erzeugen:

„Nennen sie den Ort in dem Peter Meier wohnt!"

„Wohnt Herr Meier in Berlin oder Köln?"

Strobel (1997, S. 118) gibt als Gestaltungsrichtlinie vor, dass ein System, wenn es nur eine geringe sprachliche Kompetenz hat, dem Benutzer die Verbalisierungsarbeit abnehmen soll. Das heißt, die Fragen sollen so formuliert sein, dass sie eine Minimalantwort zulassen oder es dem Benutzer leicht gemacht wird die Formulierungen aus der Frage zu übernehmen. Da die Zustimmung zu einer Frage kürzer ist als die Ablehnung oder Zurückweisung sollte eine Frage so formuliert sein, dass bei der erwarteten Antwort der Benutzer zustimmen kann. (Strobel, 1997, Kapitel 5)

Bei Alternativfragen sollten die Antwortmöglichkeiten bewusst so flexibel gehalten werden, dass auch Antwortmöglichkeiten wie „egal" oder „irrelevant" getroffen werden können. Das

10 Wenn der Benutzer z.B. nach einer Straße fragt, wird er keine Nachfrage zur Straße beantworten können.

Programm hat dann die Möglichkeit die aus Sicht des Systems beste Alternative zu wählen. (Strobel, 1997, Kapitel 1.1)

2.4 Dialoge

Es interessieren hier in erster Linie Mensch-Maschine-Dialoge. Eine Vielzahl von Eigenschaften die für zwischenmenschliche Dialoge gelten, finden in diesem Fall keine Anwendungen, aber es gibt auch einige Erkenntnisse, die man aus zwischenmenschlichen Dialogen übernehmen kann.

Im Speziellen sind hier mit Dialogen zeitlich aufeinander folgende Fragen und Antworten bzw. Sprechakte gemeint, die sich möglicherweise auch inhaltlich ergänzen. Nach Möglichkeit sollte der Dialog durch die Geprächsteilnehmer steuerbar sein, so dass die Initiative nicht nur bei einem der Teilnehmer liegt. Die Nachfragen sollten flexibel, je nach dem momentanen Kenntnisstandes des Programmes ausgewählt werden, Abkürzungen in den Dialogen, z.B. durch umfangreichere Antworten des Benutzers, sollten möglich sein.

Auch Fragebögen, sind Dialoge. Ein Feld, vor dem „Name“ steht, ist sicherlich vergleichbar mit der natürlichsprachlichen Frage „Wie heißen Sie?“ bzw. „Wie heißt die gesuchte Person?“. Nur wird in diesem Fall auch kein kompletter Satz als Antwort verlangt und die Information ist leichter zu verarbeiten. Bei diesen Fragebögen werden meistens viele Felder gleichzeitig auf dem Bildschirm dargestellt und die Aufgaben der Felder sind meistens nur stichwortartig neben den Feldern angegeben.

Ein Wizard oder Assistent, wie z.B. ein Internet-Verbindungsassistent, fragt die Informationen einzeln nacheinander ab und liefert zu jedem Feld eine längere Erklärung, was als Eingabe erwartet wird. Ganze Sätze können auch hier nicht als Antwort angenommen werden. Die Steuerbarkeit ist normalerweise bei Assistenten auch nicht gegeben, da die Strategie nach der sie Informationen sammeln vom Programmierer fest vorgegeben ist.

2.4.1 Ergänzung der Eingabe

In gewissen Fällen kann es nützlich sein, wenn dem Benutzer beim Eingeben Arbeit abgenommen wird. So können gewisse Begriffe, wie Personen- oder Ortsnamen, automatisch ergänzt werden sobald sie eindeutig erkannt sind. In der Unix-Shell kann der Benutzer durch drücken der TAB-Taste das Ergänzen der Wortes anfordern.

Wenn es in einem System notwendig ist Eingaben mit der Tastatur vorzunehmen werden solche Eingabehilfen von den Benutzern stark angenommen. Ob bei natürlicher Sprachen eine

solche Unterstützung auch von den Benutzern gewünscht wird kann nicht gesagt werden. Einige Textverarbeitungssysteme, wie StarOffice 5.2, haben solche Hilfsmittel jetzt integriert und vielleicht werden in absehbarer Zeit Untersuchungen erscheinen sie sich damit beschäftigen.

2.4.2 Turn Taking

Mit Turn Taking ist der Übergang von der gerade sprechenden Person auf den nächsten Sprecher bzw. in diesem Fall den Computer gemeint.

Zwischen Menschen funktioniert dies sehr reibungslos, nur 5% der Redebeiträge überlappen und die Pausen beim Wechsel des Sprechers sind nur wenige Zehntelsekunden lang. (Ervin-Tripp, 1979)

Es gibt die Auffassung, dass das Turn Taking des Computers bei der Kommunikation mit dem Menschen nur scheinbar ist, da es keine Initiative, keinen Themenwechsel und keine durchdachten Rückfragen von seiten der Maschine gibt (Zoeppritz, 1988).

Lässt man diesen Einwand aber mal außen vor, ist in einfachen Computerprogrammen, die mit Texteingabe arbeiten, das Turn Taking relativ einfach. Der Benutzer beendet seine Aussage durch die Eingabetaste, indem er einen Knopf auf dem Bildschirm drückt oder ähnliches. Natürlich wäre auch in einem solchen Programm eine andere Übergabe der Gesprächsführung möglich, wenn z.B. das Programm automatisch erkennt, dass der Benutzer seine Eingabe beendet hat, oder sobald das Programm die Intention des Benutzers erkannt hat. Es ist aber nicht sicher, ob solche Eingriffe in den Programmablauf für den Benutzer nachvollziehbar sind, auch wenn es Hinweise darauf gibt, dass der Benutzer damit keine Probleme haben wird.

So schlägt Susan Herring für Computer-Mediated Communication Systeme (z.B. Chat-Programme für mehrer Benutzer) vor, dass diese zukünftig als „Two-Way Communication“-Systeme[11] gestaltet werden sollten. Der Benutzer kann normalerweise mit der in dieser Situation entstehende Inkohärenz und den Verletzungen von Maximen der Kommunikation zurecht kommen, wenn ein Protokoll vorhanden ist, das ihm erlaubt den Verlauf der Kommunikation nachzuvollziehen. (Herring, 1999)

11 Was ein Benutzer schreibt, ist zur selben Zeit für alle anderen Benutzer sichtbar. Es werden alle Zeichen übertragen. Die Anderen sehen so auch schon nicht vollständige Eingaben des Benutzers und können darauf reagieren.

In der Mensch-Maschine Kommunikation ist der Bezug zu den Äußerungen des Benutzers größer, es ist im Normalfall[12] nicht damit zu rechen, dass das Programm eine eigene Intention hat und wird deswegen nur auf den Benutzer reagieren (s.o.).

Es lassen sich auch für die geschriebene Sprache Beispiele finden in denen nicht eindeutig ist, ob der Benutzer die Gesprächsführung schon übergeben wollte, da er zwar eine Aussage gemacht hat, die darin enthaltenen Anweisungen aber noch nicht zur Handlungssteuerung reichen, z.B. „`Ich suche jemanden in Berlin.`“.

Wenn man es mit gesprochener Sprache zu tun hat, ist der Wechsel des Sprechers nicht so einfach zu erkennen. Darin unterscheiden sich zwischenmenschliche Gespräche und Gespräche mit dem Computer wenig (Gespräche mit dem Computer sind in den meisten Fällen nur Experimente, die meistens nach dem „Wizard of Oz“-Szenario, bei dem einem Menschen intelligentes Verhalten der Maschine vortäuscht wird). Computer die gesprochene Sprache verstehen und nicht nur transliterieren, sind selten.

Der Mensch hat eine erlernte oder angeborene Fähigkeit zu erkennen, wann er die Gesprächsführung übernehmen kann. Verlässliche Methoden, wie der Computer diese Aufgabe erfüllen kann, müssen erst noch erarbeitet werden.

2.4.3 Shift of Intention

An dieser Stelle wird auf ein Problem eingegangen, dass nicht nur bei natürlichsprachlichen Dialogen besteht: den Intentionswechsel des Benutzers.

Wenn man ein natürlichsprachliches System benutzt, sollte der Benutzer seine Absicht ändern können, ohne eine nicht sprachliche Aktion ausführen zu müssen[13]. Am Besten wäre es wenn er seine Absichtsänderung noch nicht einmal explizit formulieren muss, sondern das System diese automatisch aus dem Verhalten des Benutzers erkennt. Ein Beispiel dafür:

`A: „Ich suche Peter Meier.“`

`B: „In welchem Ort wohnt Peter Meier?“`

`A: „Oder nein, doch lieber Klaus Schulz.“`

In diesem Beispiel ändert der Benutzer seine Intention ohne dass die vorherige Aufgabe beendet ist. Hier hat A die Änderung sprachlich kenntlich gemacht, an maschinellen

12 Eine Ausnahme bildet z.B. der Chat-bot Julia, der es ausnutzt, dass in einem Chat Programm die Gricesche Maxime der lokalen Relevanz ignoriert werden kann. Julia täuscht einem männlichen Chat-Teilnehmer vor ein weiblicher Mensch zu sein.

13 Ob dies gewünscht wird, kann man nur durch eine Evaluation klären.

Schnittstellen darf ein solches Benutzerverhalten aber nicht vorausgesetzt werden, wie in Kapitel 2.5 noch gezeigt wird.

Das Problem der unerwarteten Intentionsänderung tritt nicht nur in dem gerade angesprochen Fall auf, sondern genauso bei Programmen die eine Benutzermodellierung verwenden. Es gibt nur so lange ein korrektes Modell der Interessen des Benutzers wie diese konstant sind. Wenn dieser seine Interessen ändert, muss auch das Modell verändert werden. Dies kann geschehen, indem das neue Interesse zusätzlich zu dem bestehenden Modell gelernt wird. Das ist aber nur sinnvoll, wenn der Benutzer seine alten Interessen auch beibehält, ansonsten verfälscht es nur das Modell.

Bei einem Benutzermodell hat man es mit einer eher langsamen Veränderung der Interessen zu tun, während man bei mehreren aufeinander folgenden Datenbankabfragen einen schnellen Wechsel des Interesses hat. Sobald ein Ergebnis gefunden wurde, möchte der Benutzer die nächste Abfrage starten oder seine Suche beenden. Es ist aber nicht immer klar, wann der Benutzer das gewünschte Ergebnis gefunden hat. Eine Suche kann ergebnislos verlaufen, so dass der Benutzer mit der nächsten Suche fortfährt, ohne dass dies kenntlich ist, andererseits kann auch, nachdem ein Ergebnis ausgegeben wurde, der Benutzer feststellen, das seine Angaben falsch waren und er sie lieber korrigieren möchte. Im schlimmsten Fall führen äußere vom Programm nicht zu erfassende Faktoren (z.B. ein anderer Benutzer, Ende einer Pause, neue Aufgabenstellung durch eine andere anwesende Person) dazu, dass der Benutzer seine Absicht ändert.

Um das Problem der Intentionsänderung wenigstens teilweise lösen zu können, müssen alle erfassbaren Faktoren festgehalten und wenn möglich bewertet werden, in Bezug auf die Wahrscheinlichkeit, ob wirklich ein Wechsel stattgefunden hat. Falls dies nicht eindeutig möglich ist, ist es am sinnvollsten alle Möglichkeiten weiterzuverfolgen und anhand der Ergebnisse und dem folgenden Verhalten des Benutzers zu entscheiden, ob es doch einen Intentionswechsel gab.

2.4.4 Antizipation der Antwortmöglichkeiten

Die Erwartung einer bestimmten Antwort auf eine Frage ist ein gegenseitiges Problem. An einer NL-Computerschnittstelle ist es nicht nur der Computer, der, um das Ergebnis besser verarbeiten zu können, den Antwortbereich einschränkt, auch der Mensch stellt Annahmen über eine wahrscheinliche Antwort auf, wenn er eine Frage stellt. Der Menschen ist trotz der

antizipierten Antwort in der Lage andere Antworten zu akzeptieren. (Strobel, 1997, Kapitel 4.1.2)

Eine Erwartungshaltung über die Antworten kann bei der Vorstrukturierung eines Dialoges helfen. (Strobel, 1997, Kapitel 4.1.2) Wenn man z.B. erwartet, dass eine Frage nicht beantwortet werden kann, dann sollte diese Frage am Besten auch nicht gestellt werden.

Strobel (1997, Kapitel 4.1.2) verweist hierzu auf Frank (1979) der eine „ausgezeichnete Fortsetzung“ als „die Reaktion, die die Setzung und Voraussetzungen des Vorgängerzuges maximal akzeptiert und im Sinne der angezeigten Präferenz hinsichtlich der Fortsetzung kooperiert“ einführt.

Strobel sagt weiter, dass die „ausgezeichnete Fortsetzung“ häufig auf nur eine Möglichkeit eingeschränkt, deren Präferenz aber nicht immer angezeigt ist. Dies ist z.B. bei Examensfragen der Fall.

Es werden von Strobel (1997, Kapitel 3.4) vier Antwortmöglichkeiten unterschieden: die direkte Antwort, die die vorangegangene Frage genau beantwortet, die erweiterte Antwort, die entweder erklärend, begründend, vorgreifend oder zurückziehend ist, die einschränkende Antwort, in der die darin getroffene Entscheidung an Bedingungen geknüpft ist und zuletzt die Weiß-Nicht-Antwort.

Bevor die erwartete Antwort getroffen werden kann muss aber noch mit weiteren Nachfragen gerechnet werden. Strobel (1997, Kapitel 3.1) zeigt die möglichen Reaktionen auf initiative Züge:

1. Aspekte des Verstehens:
   I. Akustisches Verstehen
   II. Bedeutungsverstehen
   III. Referenzverstehen
2. Die Ausdruckform des initiativen Zugs:
   I. Wie dieser formuliert wurde (Wortwahl, Stil, etc.)
   II. Etwas darin Ausgelassenes, aber Erwartetes
3. Die Präsupposition(en)
4. Das Ziel und die Relevanz des initiativen Zuges
5. Sonstige Voraussetzungen für die Beantwortung
6. Die Konsequenz des Antwortbereichs
7. Die Beantwortung im eigentlichen Sinn:

I. Nur der Antwortbereich
II. Eine informativere Variante der Antwort
III. Eine einschränkende Variante der Antwort
IV. Die Aussage des Nicht-Wissens

Ein NL-System sollte jede der hier genannten Reaktionen verarbeiten können, wobei je nach Realisierung des Programmes natürlich einige Punkte fehlen und andere hinzukommen können, da Strobel nicht den Anspruch erhebt, dass die Liste vollständig ist.

In Systemen die nur mit geschrieben Sprache arbeiten, werden Fragen über das akustische Verstehen nicht benötigt, möglicherweise werden dafür aber Fragen zur Rechtschreibung nötigt.

2.4.5 Strategie der Dialogführung

Die in 2.4.4 gezeigte Liste von Strobel kann nicht nur auf die vom Computer erwarteten Antworten bezogen werden. Auch ein Mensch muss mit solchen Rückfragen rechen, deswegen bietet die Liste einen guten Ausgangspunkt für den Aufbau einer allgemeinen Dialogstrategie.

Strobel (1997, Kapitel 3.1) sagt, dass die Reihenfolge der Liste nicht zufällig ist. Wenn man eine Aussage zu einem der Punkte der Liste macht, akzeptiert man die davor liegenden Punkte. Wenn man z.B. eine Rückfrage zu den Präsuppositionen des vorausgehenden Zuges stellt, sagt man damit unter anderem aus, dass man die vorausgehende Aussage verstanden hat, sowohl akustisch als auch von der Bedeutung der Worte und der Auflösung der Referenzen her.

Ein Beispiel:

A: „Ich suche Peter Meier aus Berlin."

B: „Ich kann dort keinen Peter Meier finden. Sind sie sicher, dass er in Berlin wohnt?"

B gibt damit zu, dass er die Aussage von A verstanden hat und auch die Ausdrucksform wird von ihm akzeptiert. Nur kann er die Beziehung zwischen der Person und dem Ort nicht auflösen und stellt diese in Frage.

Punkt 5 kann man nun als Einstieg zur Rückfrage nach fehlenden Informationen sehen, die zur Beantwortung der Frage benötigt werden. Strobel versucht damit die Reaktionen zu erfassen, die nicht zu den anderen Punkten passen.

Fehlende Informationen werden durch eine Gegenfrage erbeten – wobei eine Gegenfrage wieder alle in der Liste genannten Reaktionen zulässt und die Initiative zumindest kurzfristig wechselt.

Es gibt nach Strobel (1997, Kapitel 3.2) Dialogzüge, die auf das Gesamtziel bezogen, den Dialog nicht weiterbringen, aber auf andere Weise nützlich sein können. So kann durch solche Züge das System besser auf den Benutzer eingestellt werden. Nach dem Durchlaufen dieses Dialoges wird der Ausgangsdialogszustand wieder erreicht und es kann weiter auf das Gesamtziel hingearbeitet werden.

Hierzu ein Beispiel:

```
...
A: „Können sie mir nur die Telefonnummer sagen.“
B: „Möchten sie in Zukunft nur noch die Telefonnummern erhalten?“
A: „Ja.“
...
```

Diese Dialogzüge haben nichts mit dem eigentlichen Ziel der Suche nach einer Person zu tun, aber in Zukunft kann das System effizienter arbeiten.

Es sollte dem Benutzer möglich sein, im Dialogverlauf seine bisherigen Entscheidungen zu korrigieren, z.B. nach einer bei einer Auswahlfrage gewählten Option, nocheinmal zu der Frage zurückkehren zu können und eine andere Entscheidung zu fällen bzw. in der momentanen Dialogsituation eine andere Entscheidung zu treffen und somit in einen anderen Dialogzweig zu kommen. Strobel (1997, Kapitel 3.3.2) stellt fest, dass der Dialogverlauf in den meisten Mensch-Maschine-Schnittstellen fest vorgegeben ist und nicht so flexibel geändert werden kann. Das führt zu Nachfragen aus dem Bereich 6 der Liste, der Benutzer muss sich orientieren, welche Optionen ihm nach einer Entscheidung noch verfügbar sind.

Wird von vornherein klargestellt, dass der Benutzer jederzeit seine Entscheidungen revidieren darf, kann er das System besser explorieren und damit dessen Funktionen besser erschließen, die Effizienz des Dialoges kann erhöht werden, da Rückfragen aus dem Bereich 6 seltener werden und der Benutzer wird mehr Sicherheit im Umgang mit dem System zeigen.

Wenn antizipiert werden kann, was das Dialogziel des Benutzers ist, bietet sich für das System an, z.B. durch erweiterte Antworten dem Dialog vorzugreifen und damit abzukürzen.

Strobel (1997, Kapitel 1.1) stellt fest, dass es Abkürzungen auch in Benutzeräußerungen gibt. Von daher bietet sich an, diese Abkürzungen auch zur effizienteren Dialoggestaltung zu nutzen.

Fragen sollten dahingehend formuliert werden, dass sie Abkürzungen zulassen, auch wenn dabei aufgepasst werden muss, dass nicht Antworten provoziert werden, die vom System nicht verarbeitet werden können.

Hier ein Beispiel dafür:

```
A: „Welche Person suchen sie?“
B: „Peter Meier.“
A: „In welchem Ort wohnt Peter Meier?“
B: „In Berlin.“
```

Es kann deswegen effektiver sein einen neutraleren Einstieg zu wählen.

```
A: „Guten Tag! Was kann ich für sie tun?“
B: „Guten Tag! Ich suche Peter Meier aus Berlin.“
```

Es konnten zwei Dialogzüge vermieden werden. Leider kann man bei einem freien Einstieg nicht unbedingt davon ausgehen, so dass es dort unter Umständen auch zu mehr Dialogzügen kommen kann.

Als globale Strategie sollte ein Programm versuchen, die Dialoglänge möglichst kurz zu halten. Daran ausgerichtet muss geprüft werden, ob es in einer bestimmten Situation für das System nicht möglich ist, eine Rückfrage zu vermeiden. Ob eine lokale Strategie bedeutender ist als eine globale, muss im Einzelfall geprüft werden. Wenn diese globale Strategie immer Vorrang hat, wird es z.B. nie zu den schon erwähnten eingeschobenen Dialogteilen kommen.

2.5 Besonderheiten natürlichsprachlicher maschineller Schnittstellen

Die natürlichsprachliche Kommunikation an maschinellen Schnittstellen unterscheidet sich in einigen Punkten sehr von der normalen zwischenmenschlichen Kommunikation, sowohl in der gesprochenen als auch in der geschriebenen Sprache. In diesem Kapitel werden diese Unterschiede dargestellt, wobei auch generelle Unterschiede zwischen gesprochener und geschriebener Sprache aufgezeigt werden sollen.

2.5.1 Teilmenge der natürlichen Sprache

Wie schon mehrmals erwähnt, kann an NL-Schnittstellen nur eine Teilmenge der natürlichen Sprache verarbeitet werden. Dass ein Benutzer damit rechnet und sich auch darauf einstellt, wird im Kapitel 2.5.3 „Computertalk“ noch gezeigt.

Programme sind dazu gezwungen, sich im Vokabular, der Syntax und Semantik, die verarbeitet werden kann, zu beschränken. Die Gründe liegen zum Einen darin, dass nicht alles Wissen aus diesen Bereichen systematisch erfasst ist und dem Programm zur Verfügung steht. Nicht jedes Wort der Sprache ist bekannt, da auch regelmäßig neue Worte gebildet werden und durch Komposita beliebig aus bekannten Worten neue gebildet werden können. Ein Lexikon wird niemals vollständig sein können. Die Semantik vieler Worte ist kontextabhängig und jede Bedeutung aller Worte zu erfassen ist unmöglich.

Zum Anderen kann ein großes Lexikon auch nicht effektiv verarbeitet werden, der Benutzer erwartet aber auf seine Eingabe in akzeptabler Zeit eine Antwort.

Die Ergebnisse der Evaluation zeigen im Übrigen, dass, bei längerer Verarbeitung der Eingabe, zumindest eine Sanduhr oder ein Statusbalken vorhanden sein sollten, durch die angezeigt wird, dass die Verarbeitung noch andauert.

Krause (1982) weißt darauf hin, dass syntaktische Lücken vom Benutzer besonders schwer zu beachten sind. Mit syntaktischen Lücken sind oberflächenstrukturelle Restriktionen, wie feste Wortstellung und fehlende syntaktische Variationen, gemeint. Es gelang den Benutzern in diesem Fall nicht, oder nur sehr schwer, den Fehler zu lokalisieren und durch Paraphrasierung doch noch die gewünschte Antwort zu erhalten.

Zoeppritz (1983) fordert aufgrund dieser Tatsache, dass es nicht sinnvoll ist, mit syntaktischen Varianten zu sparen. Beim USL-System wurden deswegen die Syntax gestärkt, um eine möglichst hohe Abdeckung von Oberflächenvariationen zu erreichen.

Krause (1988) zeigt, dass andere Systeme (z.B. Q&A) das Problem dadurch umgehen, dass sie die Wortstellung überhaupt nicht oder nur rudimentär beachten.

Semantische Lücken sind nach Krause (1982) hingegen leicht zu beachten, wenn sie ein ganzes Gebiet ausgrenzen. Zoeppritz (1983) zieht daraus den Schluss, „dass (dem Benutzer) das syntaktische Verhalten nicht in dem Maße bewusst wird, wie das semantische“. Im weiteren weist Zoeppritz (1983) noch darauf hin, dass die Auslassung von im System prinzipiell verfügbaren Konstruktionen bei den Benutzern zu genauso großer Verwirrung

führen kann wie syntaktische Lücken. Das Auslassen ganzer semantischer Bereiche ist für den Benutzer einsichtiger.

In längerem Umgang mit einem Programm bildet sich beim Benutzer ein Modell der sprachlichen Kompetenz eines Systems. Er wird seine Äußerungen so gestalten, dass das System sie versteht. (Strobel, 1997, Kapitel 1.2)

2.5.2 Unterschiede zwischen gesprochener und geschriebener Sprache

Es soll hier nicht auf Problemen der Spracherkennung eingegangen werden, wie der Übertragung von Phonemen auf Buchstaben oder der Erkennung von Wortgrenzen. Auch darauf, dass Dialekte und Akzente in der gesprochenen Sprache eine größere Rolle spielen als in der geschriebenen, möchte ich nicht näher eingehen. Aktuelle Spracherkennungssoftware kann ca. 10.000 Worte sprecherunabhängig auch über schlechte Verbindungen erkennen[14].

Im Folgenden soll mehr auf Unterschiede in der Wortwahl und Satzstellung und der Häufigkeit einiger der linguistischen Phänomene eingegangen werden. So soll außerdem auf Unterschiede im Verhalten von Menschen eingegangen werden, je nachdem ob sie schriftlich oder mündlich kommunizieren.

So möchte ich noch einmal auf das Turn-Taking eingehen. Dort wurde schon das Problem angesprochen, zu erkennen wann eine Äußerung beendet ist. In der geprochenen Sprache gibt es Pausen, zum einen um nachzudenken, weil einem das passende Wort nicht einfällt oder weil man überlegt was man überhaupt sagen will. Ein anderer Grund kann sein, dass man seine Äußerung abbrechen will (siehe Hauptmann und Rudnicky, 1988). Im weiteren kann man schweigen, weil man erwartet, dass der Gesprächspartner jetzt reden soll, und es mag noch andere Gründe geben, weshalb Pausen entstehen, z.B. durch äußere Einflüsse.

Wenn die Person meint, mit einem Computer zu sprechen, berichten Hauptmann und Rudnicky (1988), dass die Sprechpausen gelegentlich so lang werden können, dass selbst ein menschlicher Beobachter glaubt, die Äußerung sei beendet.

Schweigen stellt nach Strobel (1997, Kapitel 3.5) in der gesprochenen Sprache keine Nicht-Antwort da, sondern kann durchaus aufschlussreicher sein als eine schlecht formulierte Antwort, die das System nicht weiterverarbeiten kann. Schweigen kann ein Nicht-Wissen der Antwort, ein Nicht-Wissen über die Bedeutung der Frage oder eine Weigerung sein, die Frage

14 http://www.heise.de/newsticker/data/jo-14.03.02-000/

zu beantworten. Aus dem Kontext kann möglicherweise entnommen werden, um welche Art des Schweigens es sich handelt. Das Wissen um diesen Fehlschlag kann vom System ausgenutzt werden um Lösungsvorschläge zu machen.

Die gesprochene Sprache ist deutlich fehlerhafter als die geschriebene. So stellten Hauptmann und Rudnicky (1988) fest, dass 16% der zu einem Computer gesprochenen Äußerungen fehlerhaft waren, wo hingegen nur 0,7% der an einen Computer geschriebenen Äußerungen Fehler enthielten. Die gesprochenen Kommunikation mit anderen Menschen ist noch fehlerhafter, dort waren 31% der Äußerungen fehlerhaft. Fehler sind hier ganz allgemein gemeint, das können Fehler in der Orthographie, der Grammatik, der Wortwahl oder auch Störlaute sein.

Zoltan stellt für die schriftliche Kommunikation zwischen Menschen mit 4,5% Rechtschreibfehlern auch erheblich mehr Fehler fest als bei der schriftlichen Kommunikation mit Computern (nur 0,4%).

Außerdem gibt es in der gesprochenen Sprache Laute wie das „ähm“, „hm“[15] oder „oh“ die keine Bedeutung haben. Untersuchungen von Hauptmann und Rudnicky (1988) haben ergeben, dass nicht jeder Sprecher solche Störlaute verwendet. In ihrem Experiment waren es 8 von von 20 Personen. Unter den Personen, die mit einem Computer sprachen waren nur halb so viele die Störlaute erzeugten, wie unter denen, die mit einem anderen Menschen redeten und sie erzeugten nur ein Drittel der Menge gesamten Störlaute.

Hauptmann und Rudnicky stellen im weiteren fest, dass die meisten Störlaute (18) vor einer Nominalphrase auftraten, 2 in einer Nominalphrase, 6 in einer Präpositionalphrase zwischen der Präposition und dem Substantiv und 4 traten an den Grenzen anderer Phrasen als der Nominalphrase auf. Siebenmal wurden diese Laute als Teil eines falschen Beginns oder Neuanfangs eine Äußerung erzeugt. Wenn die Daten, die diesen Ergebnissen zugrunde liegen, anders ausgewertet werden, kann man feststellen, dass die meisten Störlaute vor definiten Referenzen auftreten[16] .

Durch die Möglichkeit, im Deutschen Komposita zu bilden, bin ich mir sicher, dass auch innerhalb von Komposita Störlaute auftreten können, da es bei der Suche nach der passenden Fortsetzung des Wortes zu Verzögerungen kommen kann.

15 In diesem Fall sind nur die „hm's“ gemeint, die in Pausen in einem Satz eingefügt werden und nicht die Conversational Filler.

16 In diesem Fall, wo ein Emailprogramm die Untersuchungsdomäne ist, sind es Referenzen zu Nachrichten, Dateien, Namen und Suchfeldern.

Um die Zuverlässigkeit von Spracherkennungsprogrammen zu erhöhen schlagen, Hauptmann und Rudnicky (1988) vor die Störrlaute noch genauer zu untersuchen und eine Theorie über deren mögliches Auftreten zu erstellen. Denn dass diese Laute nicht willkürlich gesetzt werden, ist für sie offensichtlich.

Zusätzlich zu dem „hm“, das als Störlaut auftreten kann, gibt es in der gesprochenen Sprache auch noch das „hm“ als Converational Filler, das je nach seiner Intonation und dem Zeitpunkt zu dem es geäußert wird eine Bedeutung wie Zustimmung oder Ablehnung hat (Ronthaler, 2000). Diese Bedeutung kann aber von einem Computer nur schwer erschlossen werden. Es ist zu überdenken, ob es sinnvoll ist, dass ein Computer diese „hm's“ selber anwendet.

Nach Zoltan et al. (1982) brauchen Benutzer zur Lösung einer Aufgabe in gesprochener Sprache ungefähr doppelt mal soviel Zeit wie in der schriftlichen Kommunikation. In Hauptmann und Rudnickys (1988) Untersuchungen konnte das nicht bestätigt werden, dort waren die Bearbeitungszeiten für die mündlich zu lösenden Aufgaben ungefähr genauso lang wie die in der schriftlichen Bedingung.

In Zoltans Untersuchung benutzten die Versuchspersonen wenn sie sprachen ungefähr dreimal so viel Worte, bei Hauptmann waren es nur ca. doppelt so viel wie wenn sie tippen mussten.

Sowohl bei Hauptmann als auch bei Zoltan waren die Äußerungen in den gesprochenen Szenarien ungefähr doppelt so lang wie in den Schriftlichen.

Zoltan gibt zusätzlich an, dass sowohl beim Reden wie dem Schreiben dieselbe Anzahl an Äußerungen gebraucht wurden.

Die Ergebnisse zu den letzten beiden Punkte, Länge der Äußerung und Anzahl der Äußerungen, verwundern Zoltan. Mehrere andere Studien[17] haben angeblich ergeben, dass bei gesprochener Kommunikation die Länge der Äußerungen gleich bleibt und dafür mehr Äußerungen verwendet würden. Mögliche Gründe dafür sieht Zoltan darin, dass mit einem Computer kommuniziert wird, in der Aufgabe die in diesem Experiment verwendet wurde (Kontoführung) und zuletzt in der Art wie der Computer die Ergebnisse präsentierte (auf einem Bildschirm).

Hauptmanns Ergebnis zur Länge der Äußerungen unterstützt die Vermutungen zur Kommunikation mit dem Computer und zur Präsentationsart.

17 Zoltan nennt diese Studien leider nicht.

Eine gesprochene Aussage kann nicht korrigiert werden wie eine schriftliche, indem man sie löscht oder streicht. Das Gesagte wird im Normalfall auch zur selben Zeit vom Hörer wahrgenommen werden. Eine Korrektur muss durch zusätzliche Äußerungen sprachlich kenntlich gemacht werden. Diese Änderungen müssen zusätzlich bei Systemen beachtet werden, welche die gesprochene Sprachen verstehen sollen.

Bei gesprochener Sprache bemerken Hauptmann und Rudnicky (1988), dass zu einem geringen Anteil die Aussagen Wortwiederholungen, nicht auf die Aussage bezogenen Wörter und andere Startprobleme enthielten. In der geschriebenen Sprache trat nichts vergleichbares auf.

In der gesprochenen Sprache gibt es Nachschaltungen, wie „ja?“, „verstehst du?“, „weißt du?“, „oder?“ oder „oder was?“, die die Rezeption der Äußerung durch den Hörer kontrollieren sollen. (Strobel, 1997, Kapitel 2.3.1)

Auch nur beim Gesprochenen treten Äußerungen auf, die nicht in Beziehung zur eigentlichen Aufgabe stehen. Bei Gesprächen mit anderen Menschen sind 4% der Äußerungen dieser Art, wenn man glaubt zu einem Computer zu sprechen, sind es immerhin noch 0,5%. (Hauptmann und Rudnicky, 1988)

Aus seinem Experiment an einem Mailprogramm berichten Hauptmann und Rudnicky (1988), dass die Versuchspersonen, die schreiben mussten, sehr stark auf Abkürzungen, die bei Mailprogrammen üblich waren, zurückgriffen, wo hingegen in den gesprochenen Bedingungen keine Abkürzungen verwendet wurden. Der hohe Anteil an Abkürzungen von 14,5% ist aber wohl nur dadurch zu erklären, dass die Versuchspersonen schon lange mit dem Umgang mit herkömmlichen Mailprogrammen vertraut waren.

Aber auch in der normalen geschriebenen Sprache gibt es viele geläufige Abkürzungen wie „Str.“ für Straße oder „z.B.“ für zum Beispiel, die häufig verwendet werden.

Ellipsen zählen zu den häufigeren Konstruktionen in der gesprochenen Sprache (Block und Schachtl, 1995), aber sie treten auch unter schriftlichen Szenarien auf, wenn der Benutzer weiß, dass ein System in seinem Sprachverstehen nicht beschränkt ist (Krause, 1995). In Frage-Antwort Paaren muss vor allem mit ihnen gerechnet werden (Strobel, 1997), und das wohl vollkommen unabhängig vom Eingabemedium.

Block und Schachtl (1995) stellen ebenfalls fest, dass in der gesprochenen Sprache unvollständige Sätze häufiger auftreten.

Parenthese ist sicherlich auch in der gesprochenen Sprache deutlich häufiger (Block und Schachtl, 1995) , auch wenn sie in der geschriebenen Sprache vorkommen kann.

2.5.3 Computertalk

Krause (1995) sieht die Kommunikation mit dem Computer als eine spezielle Vereinfachung der normalen Sprache, vergleichbar „baby talk“ oder „foreigner talk“[18]. Er meint, in den USL-Protokollen und DICOS-Test, die in Kapitel 3.1 und 3.6 noch vorgestellt werden, Belege dafür zu finden

Nach Krause ist die Grundidee dabei, dass die Änderungen in der Sprache vorhergesagt werden können. Die für Computertalk bedeutendensten Änderungen sind Simplification und Clarification[19].

Simplification bedeutet, dass es Reduzierungen an der Oberflächenstruktur der Äußerung gibt. Manche Grammatikregeln und Teile der lexikalischen Wissens werden weggelassen um die Verständlichkeit für den Dialogpartner zu erhöhen, wenn z.B. dessen sprachliche Fähigkeiten begrenzt sind. (Krause, 1995) In diesem Fall sind die von Krause vorgebrachten Vereinfachungen fehlende Propositionen, fehlende Verben und die fehlenden Überleitungen zwischen den Satzteilen.

Das Ziel von Clarification ist es, die Verständlichkeit der Sprache zu erhöhen, ohne die Vereinfachungen in der Oberflächenstruktur. Dazu wird die Redundanz der Mitteilung erhöht, indem Material zum Satz hinzugefügt wird, das normalerweise nicht geäußert würden. (Krause, 1995)

Wenn der Benutzer glaubt, der Gesprächspartner sei ein Computerprogramm, macht er Annahmen über die besondere Kompetenz und beschränkten sprachlichen Fähigkeiten des Systems. Simplification und Clarification treten als Folge dieser Einschätzung auf. (Krause, 1995)

Krause gibt für Simplification und Clarification Beispiele aus den USL-Protokollen. Eines seiner Beispiele „who live in 'the city' los angeles“ entspricht dem hier gezeigten Phänomen einem Beispiel von Zoeppritz (1995) über den Einfluss der Paraphrasierung in USL:

```
Eingabe: „Exportiert Spanien“
```

18 Die Art und Weise in der man mit Kleinkindern und Ausländern redet.
19 Die Begriffe entnimmt Krause der Register Theorie von Fergusson und DeBose (1977)

```
Paraphrasierung: „Führt der Exporteur Spanien aus“
```

Dabei ist das „Exporteur“ nicht notwendig und nur zu Klarstellung (Clarification) vom System hinzugefügt worden. Zoeppritz (1995) beschreibt eine Konvergenz, die Gewohnheiten der Gesprächspartner zu übernehmen, so dass von den Benutzern das unnötige „Exporteur“ bei späteren Anfragen hinzugefügt wird.

Krause (1995) behauptet, fast ein Drittel der Fragen in den USL Protokollen wären Abweichungen von der normalen Sprache. Wenn man aber die Auswirkungen der Paraphrasierung mit einbezieht, sind meines Erachtens die USL-Protokolle keine gute Quelle sind, um die Clarification Annahmen zu belegen.

Die von Krause (1995) vorgestellten Ergebnisse aus DICOS zeigen, dass zumindest Ellipsen an beschränkten natürlichsprachlichen Schnittstellen deutlich weniger verwendet werden.

Block und Schachtl (1995) vertreten die Meinung, dass Benutzer unter „stressy“ Bedingungen dazu neigen, die Sprache zu vereinfachen, und das Bewusstsein der Kommunikation mit einem Computer stellt eine solche Bedingung dar. Die Vereinfachungen, die sie nach Weiss (1975) angeben, entsprechen nicht ganz den von Krause genannten Einschränkungen, sollen aber aus meiner Sicht dasselbe Problem formalisieren.

Wenn man von der Idee von Block und Schachtl ausgeht, müsste man davon ausgehen können, dass bei einer Gewöhnung an eine Kommunikation mit dem Computer die „stressy“ Einflüsse verschwinden und sich die Kommunikation mit dem Computer verändert. Insoweit kann man Krauses Computertalk-Theorie vielleicht nur als eine Momentaufnahme sehen, die zwar auf die künftige Gestaltung von NL-Schnittstellen Einfluss haben sollte, aber an Bedeutung verlieren wird, je erfolgreicher und akzeptierter diese Schnittstellen werden.

2.5.4 Weitere Unterschiede zur Kommunikation zwischen Menschen

Die normale schriftliche Kommunikation zwischen Menschen unterscheidet sich von dem, was in der Schule gelehrt wird. Aber Gespäche mit einem Computer sind nicht ganz so ungeregelt wie zwischenmenschliche Kommunikation.(Hauptmann, Rudnicky, 1988)

Wie auch schon in den von Zoltan (1982) berichteten Ergebnissen (siehe Kapitel 2.5.2) ist das, was Menschen sich gegenseitig schreiben recht fehlerhaft. Glücklicherweise sind Menschen in der Kommunikation mit Maschinen sorgfältiger und produzieren weniger Fehler.

Die Kommunikation mit der Maschine ist deutlich kürzer. Es werden weniger Höflichkeitsfloskeln verwendet. Der Benutzer fühlt sich gegenüber einem Computer nicht zu

kausalen, konsekutiven oder adversativen Äußerungen verpflichtet und Reduktionen und Ellipsen sind zahlreicher. (Strobel, 1997, S.15f)

Die Behauptung zu Ellipsen steht im Widerspruch zu der in Kapitel 2.5.2 geäußerten Behauptung Krauses, dass an beschränkten NL-Schnittstellen Ellipsen weniger auftreten. Es läßt sich dadurch erklären, dass Strobel vorwiegend Frage-Antwort-Dialoge untersucht hat, in denen teilweise sogar beabsichtigt wurde, durch die Fragestellung minimale Antworten zu erzeugen.

Krause (1995) stellt zusätzlich fest, dass komplexe Satzstrukturen und indirekte Fragen in der Kommunikation zwischen Menschen deutlich häufiger auftreten als in der Situationen, wo der Benutzer glaubt, mit einem Computer zu kommunizieren. Der Benutzer passt sich in diesem Punkt sogar an das System an, falls dieses beschränkte sprachliche Fähigkeiten hat.

Bei der gesprochenen Sprache gehen noch mehr Informationen beim Übertragen der Sprache verloren als beim Schriftlichen. Keine Spracherkennung kann, meines Wissen, die Intonation einer Äußerung verwerten. So dass z.B. bei den schon angeprochenen „hm's“, ein Großteil ihrer Bedeutung beim Übertragen verloren geht. Die Intonation kann bei der Spracherkennung sogar zu einem Problem werden, wie Hauptmann und Rudnicky (1988) berichten, wenn nämlich durch eine Fehlerkennung der Benutzer gezwungen ist, ein Wort zu wiederholen. Das Wort wird dann überbetont gesprochen, um es deutlicher zu machen, die Spracherkennungsprogramme sind damit aber überfordert.

2.5.5 Dialogsprache

Strobel (1997, Kapitel 4.1) sieht Frage und Antwort als eine Einheit. Die Frage ist ein 'unvollständiger' Satz der eine Antwort 'fordert'. Die Frage kann durch ihre Form gewisse Antworten präferieren oder dispräferieren und so in der Antwort Reduktionen erzeugen (vgl. auch Kapitel 2.3).

Durch diese enge Beziehung des Fragesatzes und des Antwortsatzes wäre eine Dialogsyntax bedenkenswert, die es aber nach Strobel (1997, Kapitel 3.6) nicht gibt. Es muss immer noch die Syntax der einzelnen Sätze verarbeitet werden statt der des gesamten Dialogs.

Zu der Sprache in Dialogen ist noch die Tendenz zu erwähnen, Äußerungen des Gesprächspartners zu übernehmen. Dies geschieht möglicherweise aus ökonomischen

Gesichtspunkten, da es einfacher ist, Formulierungen zu übernehmen, als eigene zu finden. (Strobel, 1997)

Durch ein „Recipent Design“ sollen Äußerungen auf den Adressaten und sein Wissen zurechtgeschnitten sein, ein rasches Verstehen soll so ermöglicht und Verständigungsproblemen vorgebeugt werden. (Bergmann, 1994)

2.6 Auswirkung der im Telefonauskunftsprogramm gewählten Benutzerschnittstelle

In dem Telefonauskunftsprogramm, das in Kapitel 4 noch gezeigt wird, wird eine Schnittstelle benutzt die geschriebene Sprache verarbeitet. Es gibt ein Textfeld in dem der Benutzer seine Eingabe machen kann und ein größeres Textfeld in dem der bisherige Dialogverlauf erfasst wird. Eine genauer Beschreibung der Schnittstelle ist in Kapitel 4.2.1 zu finden. Hier soll nun aufgezeigt werden, welchen Einfluss diese Schnittstelle möglicherweise auf das Benutzerverhalten haben kann.

Susan Herring beschreibt für ein Chatprogramm die Notwendigkeit einer History-Funktion, die es einem Benutzer ermöglicht, sich den Verlauf der Unterhaltung noch einmal zu vergegenwärtigen. Der Bedarf der History entsteht dadurch, dass Benutzer in solchen Diskusionen manchmal zusammenhanglose Äußerungen machen, die aber, wenn man etwas weiter im Gespräch zurückgeht, doch einen Bezug haben.(Herring, 1999)

In diesem Fall haben wir nur einen Dialog mit zwei Teilnehmern, von denen der eine ein Computerprogramm ist, das nicht absichtlich solche Inkohärenzen erzeugt. Trotzdem kann es natürlich vorkommen, dass es dem Benutzer so vorkommt, als ob Programm sich unerwartet verhalten würde. Das kann dann zum Beispiel daran liegen, dass die Eingabe des Benutzers falsch interpretiert wurde oder das System einen anderen Informationsstand hat als der Benutzer dachte. Die History kann ihm hier helfen, seine eigenen Äußerungen noch einmal zu kontrollieren und die Reaktion des Systems nachzuvollziehen.

Bei der Untersuchung von Protokollen sind Herring Situationen aufgefallen, in denen Benutzer auf Fragen geantwortet haben, die bis zu 50 Schritte vorher gestellt wurden. (Herring, 1999)

Bei der gegenwärtigen Konzeption des Programmes sind nicht einmal so lange Dialoge zu erwarten. Trotzdem scheint es mir, dass, wenn das System einmal Referenzen korrekt auflösen können soll, man durch die History Funktion mit einem sehr großen Kontext rechnen muss, in

welchem Referenzen und Anweisungen noch aktiv sind, da die Gedächtnisspanne des Benutzers durch die History vergrößert wird.

3 Überblick über existierende Systeme

In diesem Kapitel werden ein paar ausgewählte Systeme vorgestellt, die im Bereich der natürlichsprachlichen Datenbankabfrage bzw. Informationsvermittlung angesiedelt sind.

Das User Specialty Languages System ist hauptsächlich eine Datenbank, deren Besonderheit die Verwendung natürlicher Sprache ist.

Q&A ist eine Textverarbeitung und Datenbank mit einer zusätzlichen natürlichsprachlichen Datenbankschnittstelle.

Ask Jeeves ist eine Internetsuchprogramm mit einem natürlichsprachlichen Interface.

C@ll + Find ist eine Suchmaschine, die über Sprache gesteuert wird. Deswegen werden die Informationen über einen Dialog eingegeben.

COMODA ist ein natürlichsprachliches Hypertextsystem.

DICOS ist eine Studie, die Unterschiede in den Äußerungen eines Benutzers im Mensch-Mensch Dialog gegenüber dem Mensch-Maschine-Dialog herausfinden will.

3.1 User Specialty Languages

„User Specialty Languages“ ist ein natürlichsprachliches anwendungsunabhängiges Datenbanksystem. Das USL wurde seit 1970 am wissenschaftlichen Zentrum der IBM in Heidelberg entwickelt und ab 1991 von IBM unter der Produktbezeichnung „Language Access“ vertrieben und dann schon nach kurzer Zeit eingestellt.

USL wurde so entwickelt, das es mit mehreren Sprachen genutzt werden konnte, und Module für Deutsch, Englisch und Spanisch entwickelt wurden. (Zoeppritz, 1983)

In einem älteren Artikel von Zoeppritz (1983) wird das User Specialty Languages System beschrieben. Das Programm verwandelt eine natürlichsprachliche Eingabe in einen oder mehrere SQL-Ausdrücke . Es gibt ein anwendungsunabhängiges Vokabular von ca. 450 Wörtern als Vollformen[20]. Der Benutzer kann zusätzlich noch ein Vokabular für seine Anwendung passend zur benutzten Datenbank definieren. Namen, die in der Datenbank stehen, brauchen nicht extra definiert zu werden.

Alle Verben, Substantive und Adjektive werden als Relationen aufgefasst. Das System ist von vornherein so entworfen worden, dass mehrere Relationen in einer Abfrage verbunden werden können. Damit es aber nicht zu unnötigen Konflikten mit der Datenbank kommt, werden die für die Analyse notwendigen Beziehungen als virtuelle Relationen über die Relationen der Datenbank definiert. Beim Übersetzen ergeben sich Verbunde zwischen den

20 Die Quelle hierzu ist von 1983, da das System bis 1991 weiterentwickelt wurde, kann sich noch einiges geändert haben.

virtuellen Relationen, die dann wo möglich auf Basisrelationen zurückgeführt werden können. (Zoeppritz, 1983)

Datenbezogene oder inhaltliche Nachfragen vom System waren nicht vorgesehen, aber falls die Anfrage nicht eindeutig war, wurden dem Benutzer natürlichsprachliche Paraphrasen der unterschiedlichen generierten SQL-Ausdrücke präsentiert. Aus diesen Pharsen konnte der Benutzer dann auswählen, welche Anfrage er stellen wollte. Das Programm erzeugte aus der mehrdeutigen natürlichsprachlichen Eingabe des Benutzers die dieser Eingabe entsprechenden SQL-Ausdrücke. Dies geschieht ohne in der Datenbank zu kontrollieren, welche der Anfragen überhaupt erfolgsversprechend wären. Aus diesen SQL-Ausdrücken werden dann wieder natürlichsprachliche Sätze erzeugt. Die Paraphrasierung benutzt teilweise eigenartige Satz- und Wortkonstruktionen, nur um eine eindeutige natürlichsprachliche Aussage zu präsentieren. Diese Konstruktionen sind nicht immer die vom System am Besten zu verarbeitenden Sätze, leider muss mit einer Adaptation des Benutzers an die Paraphasen gerechnet werden. (Zoeppritz; 1995 und Zoeppritz, 1983)

User Specialty Languages zeigt, dass natürlichsprachliche Datenbanken möglich sind und auch einige Vorteile gegenüber herkömmlichen Datenbanken bieten. In den Studien zu diesem System werden einige der angeblichen Nachteile von natürlicher Sprache gegenüber strukturierten Abfragesprachen widerlegt.

Nach Krause (1988) ist USL „auch heute noch weltweit gesehen das am extensivsten evaluierte natürlichsprachliche Frage-Antwort-System. Es wurde mit mehr als 100 Benutzern getestet, die über 12.000 Anfragen produzierten.

Aus meiner Sicht ist der nicht anwendungsspezifische Ansatz von User Specialty Languages unter Umständen eine der Schwächen des Konzepts. Das System ist gut durchdacht, da es aber als eine allgemeine Datenbank gedacht ist, muss der Benutzer sich über die Möglichkeiten einer Datenbank im Klaren sein. Sogar wenn das Vokabular an die momentane Anwendung angepasst werden kann, sind die Möglichkeiten mehr oder weniger von der Funktionalität einer Datenbank abhängig. Der Benutzer hat zwar eine natürlichsprachliche aber keine natürliche Interaktion mit dem System.

Natürlich ist dieses Argument ein wenig hinfällig, da das System nun einmal als Datenbank konzipiert worden ist, aber es ist meiner Meinung nach vielleicht einer der Gründe für das schnelle und bedauerliche Scheitern von Language Access und für das mangelnden Interesse der Besucher auf der CEBIT 1992[21].

21 Magdalena Zoeppritz beschreibt, dass Language Access zusammen mit einem Spracherkennungs-Prototypen auf der CEBIT 1992 präsentiert wurden. Während die Sprache-zu-Text-Umwandlung von den meisten Besuchern als Sensation betrachtet wurde, überraschten die Fähigkeiten von Language Access nur

Ein weiteres Manko ist, dass die Ausgabe nicht ebenfalls natürlichsprachlich ist, was auch wieder mit der Konzeption als Datenbank und den Möglichkeiten zur Ausgabe auf einem Bildschirm zusammenhängt. Die Illusion einer natürlichsprachlichen Kommunikation kann so nicht entstehen.

3.2 Q&A

Q&A wurde 1985 von Symantec als Textverarbeitungs- und Datenbanksystem veröffentlicht. Kurze Zeit später erschien eine deutsche Version unter dem Namen F&A. Nach der Version 5 für DOS wurde 1995 die Weiterentwicklung eingestellt.

Die Datenbank besaß von Beginn an eine zusätzliche natürlichsprachliche Schnittstelle. Die NL-Schnittstelle war aber nie als einziger Zugang zur Q&A Datenbank gedacht. In Q&A gab es immer die Möglichkeit, problemlos zu einer „guten, leicht bedienbaren formale Abfragemöglichkeit“ (Krause, 1988) zu wechseln.

Später kam auch noch ein menübasiertes natürlichsprachliches Interface nach der von C.W.Thomson in NL-Menu entwickelten Technik hinzu. (Arz, Flassig und Stegentritt, 1995)

Da es sich um ein kommerzielles Produkt handelt, sind Informationen über tatsächliche Realisierung der NL-Schnittstelle nicht verfügbar. Krause (1988) stellt einige Studien vor, die sich mit Q&A beschäftigt haben und versucht, aufgrund praktischer Erfahrungen mit dem System etwas über die Realisierung zu sagen. Die Untersuchungen Krauses beziehen sich über die damals aktuelle Version 1.1, in den folgenden Versionen bis 1995 kann sich natürlich noch einiges geändert haben.

In Q&A gibt es, wenn überhaupt, nur eine rudimentäre syntaktische Komponente. Wortstellung und ähnliches werden nicht beachtet. (Krause, 1988)

Das Ausgabeformat kann vom Benutzer verändert werden. Das Programm besitzt aber keine ausgereiften Heuristiken oder Partnermodellierung, um eine optimale Ausgabe zu erreichen. Vielmehr wird über globale Voreinstellungen und vordefinierte Phrasen, die erlernt werden müssen, die Ausgabe verändert. So gibt es z.B. global voreingestellte „identification columns“ die normalerweise bei jeder Abfrage ausgegeben werden und mit der Phrase „with no identification column“ oder der Akürzung „WNIC“ für die aktuelle Anfrage ausgeblendet werden können. Bei where-Fragen werden normalerweise alle Spalten ausgegeben, die als „location“ definiert sind. Die Phrasen, die für die Änderung des Ausgabeformates gelernt werden müssen, weisen alle Nachteile[22] einer formalen Anfragesprache auf. (Krause, 1988)

Wenige. (Zoeppritz, 1995)

22 Siehe auch Kapitel 2.2.1

Es kann nur nach Daten gefragt werden die in einer Tabelle stehen. Eine Join-Operation mit der die Spalten mehrerer Tabellen verknüpft werden können, ist nicht möglich. (Krause, 1988)

Komplexere Anfragen, in denen mehrere Anfragen verknüpft werden, sind nicht möglich, sondern müssen einzeln nacheinander gestellt werden[23](Krause, 1988). Krause liefert dazu ein Beispiel aus dem Q&A Handbuch (Symantec 1986: IA – 72):

Es funktioniert nicht:

```
What's John's manager's age?
```

oder

```
What ist the age of the manager of John?
```

Diese Anfrage muss durch folgendes ersetzt werden:

```
Who is John's manager?
```

Antwort: Sarah.

```
What's Sarah's age?
```

Krause (1988) gibt der NL-Schnittstelle von Q&A keine gute Bewertung, sondern sieht darin eher eine Spielerei. Dem Paket aus Textverarbeitung und Datenbank bescheinigt er 1988 aber, dass es das Geld wert ist.

3.3 Ask Jeeves

AskJeeves ist meines Wissens eines der wenigen aktuellen Projekte im Bereich von Datenbanken mit natürlichsprachlicher Schnittstelle. Die Firma wurde 1996 gegründet und bietet einen Internetsuchdienst an, dessen Schnittstelle so designt ist, dass der Benutzer seine Anfrage als Frage stellt.

Der Datenbestand, über den „Ask Jeeves“ arbeitet, ist dreigeteilt:

- Ein von Hand angelegtes Verzeichnis von Internetseiten, bei dem jeder Eintrag mit einigen Fragen (oder Schlagworten), die zu dieser Seite passen, versehen wurde.
- Ein von Teoma Search Technologies entwickeltes System, dass die beliebtesten Internetseiten zu einem Thema liefern soll.
- Ein Metasuchprogramm, dass die Anfrage teilweise vereinfacht an andere Internetsuchmaschinen weiterleitet und die besten Ergebnisse zurückliefert.

Die Frage-Technologie wurde von Ask Jeeves auch an andere Anbieter wie Altavista verkauft. Mit JeevesOne wurde eine eigenständige Suchmaschine für Firmennetzwerke entwickelt, die automatisch Dokumente klassifizieren kann und auf diesem Datenbestand die

23 Was man auch als den Ansatz einer Dialogkomponente sehen kann, obwohl es wünschenswerter wäre, nicht auf mehrere Fragen für die Anfrage angewiesen zu sein.

Frage-Technik anbietet. Eine eigene Klassifikation von Dokumenten gibt es nach den Informationen auf der Webseite von Ask Jeeves[24] bei der Internetsuchmaschine nicht.

Nach Petreley (1999) vergleicht das System, bei der Suche im eigene Datenbestand, die eingegebene Frage mit den in der Datenbank gespeicherten Fragen und gibt die ähnlichsten Fragen aus.

Nach meinen Tests mit dem System habe ich den Eindruck, dass der einzugegebende Fragetext ignoriert wird und nur Stichworte aus der Frage extrahiert werden. So ergeben die Fragen

„Santa Claus"
„Who is Santa Claus?"
„Where does Santa Claus live?"
„What does Santa Claus bring?"

dasselbe Ergebnis, besser gesagt werden einem vom System die selben Nachfragen gestellt:

„Where is Santa Claus right now?"
„What is the history of Santa Claus?"
„Where can I buy [Santa figures] online?"
„Where can I find encylopedic biographical resources on [Santa Clause]?"
„Where can I find information about the movie [The Santa Clause]?

Die in den eckigen Klammern angegeben Worten sind Pull-Down-Menüs mit einer Auswahl an ähnlichen Worten, aus denen der Benutzer wählen kann. Die Alternativen für „Santa figures" sind Gegenstände die etwas mit Weihnachten zu tun haben. Für „Santa Claus" sind eine Vielzahl Namen die mit S beginnen wählbar und statt „The Santa Clause" können Filme die sich mit Weihnachten beschäftigen ausgewählt werden.

Stellt man auch exakt eine der Fragen, die als Antwort ausgegeben werden, werden einem doch die selben 5 Fragen als Ergebnis präsentiert.

Die hier gemachten Aussagen gelten nur für den von Ask Jeeves selbst bereitgestellten manuell zusammengestellten Links, da gerade für diesen Datenbestand die besten Ergebnisse zu erwarten sind.

Aus meiner Sicht ist es enttäuschend zu sehen, dass das Wissen über Fragen nur dazu genutzt wird Stichworte zu extrahieren. Syntaktisches und semantisches Wissen werden anscheinend nicht benutzt.

Eine Dialogkomponente, wie sie in dieser Arbeit angedacht ist, gibt es bei Ask Jeeves nicht. Die Umformulierung ist nur wegen der schlechten linguistischen Fähigkeiten des Systems nötig und stellt in vielen Fällen einen unnötigen Interaktionschritt dar.

24 www.ask.com

3.4 C@ll + Find

Das Programm C@ll + Find wurde Ende 2001 von DeTeMedien und Immobilienscout24 veröffentlicht. Die Entwicklung übernahm T-Nova. Es handelt sich dabei um eine telefonisches Immobiliensuchprogramm für Berlin. Die Daten werden von Immobilienscout24 zur Verfügung gestellt. Das Programm befindet sich noch im Versuchsstadium und ist noch auf 6 Monate begrenzt.

Die Eingabe erfolgt über eine Spracherkennung und, falls der Benutzer es wünscht, auch über den Ziffernblock der Tastatur.

Die Sprachausgabe des Systems besteht zum größten Teil aus vorher aufgenommenen Sätzen. Nur dort, wo Benutzereingaben bestätigt werden sollen und bei der Ausgabe der Ergebnisse, wird eine synthetische Sprachausgabe eingesetzt. Der Benutzer kann sich über Telefon, per Fax oder über SMS darüber benachrichtigen lassen, dass neue Ergebnisse gefunden wurden. In der von mir gewählten Option SMS waren in der Benachrichtigung aber keine Informationen über das Suchergebnis enthalten.

Die Fragen des Systems sind hauptsächlich Auswahlfragen[25] oder Ja-Nein-Fragen. Bei Fragen, die einen komplexere Antwort verlangen, ist in der Frage die Formulierung der Antwort weitestgehend vorgegeben, z.B. „Nennen sie jetzt bitte die gewünschte Grundfläche in Quadratmetern in der Form von ... bis ...“

An einigen Stellen kann die Auswahl der Fragen, die zusätzlich gestellt werden, beeinflusst werden (z.B. „Suchen sie ein Haus oder eine Wohnung?“). Die eingestellte Sequenz von Fragen ist durch den Benutzer in ihrer Reihenfolge nicht veränderbar. Die einzige Ausnahme stellen Fehlermeldungen dar, die je nach Bedarf natürlich im Dialog auftauchen können. Meistens waren Fehlermeldungen nur Wiederholungen der letzten Frage.

Der Dialog kann nicht durch komplexere Äußerungen, wie „Ich suche eine 3 Zimmerwohnung in Wedding“, verkürzt werden, die Fragen lassen dies durch ihre Formulierung auch nicht zu.

Der Dialog entspricht hier bei einem Teil der Fragen einem Formular, in dem Felder auszufüllen sind und beim anderen Teil einem Auswahlmenü, aus dem der Punkt gewählt werden muss, den man bearbeiten will. Die Initiative für Fragen liegt hier ausschließlich beim Programm. Der Dialog ist der Ersatz für die fehlende grafische Oberfläche; die Fragen sind mehr oder weniger nur eine 1:1 Modellierung der dortigen Gestaltungsmittel. An den Möglichkeiten eines natürlichsprachlichen Dialogs hat man sich nicht orientiert.

25 Z.B.:„Entscheiden sie sich nun bitte zwischen Suchprofil, Umzugservice oder nennen sie ihre Kundennummer.“

Das Turn-Taking wird durch Anweisungen beeinflusst. Es wird in den Erläuterungen zum System darauf hingewiesen, dass der Benutzer schon reden kann, während das System noch spricht.

Den Aussagen werden scheinbar nur Stichworte entnommen, sofern nicht durch die Fragestellung nur einzelne Worte zu erwarten sind.

Auf Anfrage teilte mir der Hersteller mit, dass der Sprachdialog leistungsfähiger sei, als es den Anschein habe: Es ist möglich in Sätzen zu sprechen, teilweise können mehrere bedeutungstragende Elemente zugleich eingegeben werden, und es ist eine Vielzahl alternativen Äußerungen zugelassen.

Pressemitteilungen zu dem System, in denen einige spärliche Informationen gegeben wurden, sind leider schon von den Internetseiten entfernt.

Die Ausführungen hier beruhen also zum größten Teil auf meinen eigenen Erfahrungen mit dem System.

Es war mir trotzdem wichtig, dieses System vorzustellen, da es sich vom Konzept her deutlich von den anderen vorgestellten unterscheidet. Sicherlich gibt es auch andere elektronische Datenbanksysteme, die möglicherweise die selben Kriterien erfüllen, aber mir war es wichtig, ein System zu zeigen, das auch mit Spracheingabe komplizierterer Ausdrücke als Ja/Nein zurechtkommt und die dafür angewendeten Strategien zu zeigen.

3.5 COMODA

COMODA steht für COnversational MOdel of Database Access. Das Programm wurde von Patrick und Whalen Ende der '80 Jahre entwickelt. In diesem System soll es möglich sein, mit natürlichsprachlichen Fragen durch eine hypertextähnliche Datenbank zu navigieren.

Das übergeordnete Ziel der COMODA-Entwicklung sollte ein System sein, das als öffentliches Informationssystem genutzt werden kann (Patrick, Jacques-Locmelis und Whalen, 1993). Ein System das über AIDS informierte, wurde ebenfalls kommerziell vertrieben (Patrick und Whalen, 1992).

Die Datenbasis, auf der COMODA arbeitet, ist austauschbar. Patrick und Whalen (1992) geben ein AIDS-Informationssystem und eine Datenbank über „The rules and terms of baseball“ an.

Der Benutzer kann dem System Fragen stellen und erhält dann eine Antwort, die entweder die Frage direkt beantwortet oder die Informationen enthält, die der Beantwortung der Frage

am nächsten kommen. Dazu werden Muster in den Fragen gesucht und verglichen mit Informationen, die an den Antworten annotiert sind. Die Muster können einzelne Worte sein, aber auch Phrasen und ganze Satzteile. Die annotierten Informationen müssen dazu vollständig sein und auch Synonyme erfassen können. (Patrik et al., 1993)

Eine lexikalische Komponente, die z.B. automatisch die Synonyme auflösen kann, gibt es nach den Beschreibungen nicht, Patrik et al. (1993) weisen darauf hin, dass alle Muster bekannt sein müssen. Hierin sehen sie kein Problem, da Benutzer in der Lage sind, die möglichen Einschränkungen beim Nachfragen zu erkennen, sofern es sich um ein „natürliches“ Thema handelt. Die Benutzer zeigen auch Verständnis, wenn eine Frage die außerhalb eines „natürlichen“ Themas liegt, nicht beantwortet werden kann.

Wie genau man ein „natürliches“ Thema erkennen kann, können Patrik und Whalen nicht angeben. In ihren Versuchen zeigte sich aber, dass die Krankheit AIDS von den Benutzern als ein solches empfunden wurde und nur wenige Fragen außerhalb des beantwortbaren Bereiches gestellt wurden. Bei „The rules and and terms of baseball“ war dies nicht der Fall, da immer wieder Fragen zu aktuellen Spielern und Statistiken gestellt wurden.

Die Daten sind nach Patrick und Whalen (1992) so organisiert, dass es von einer Information Verknüpfungen zu anderen Texten gibt, die inhaltlich Bezug zu der ausgegebene Information haben. Zusätzlich sind die Informationen thematisch zusammengefasst. Diese thematische Ordnung ist hierarchisch organisiert, an jedem Knoten sind Muster und Texte gespeichert. Wenn in dieser Hierarchie gesucht wird, werden die Muster der Frage abgearbeitet bis ein Text gefunden wird, der der gesuchten Information am nächsten kommt.

Wenn der Benutzer nun eine Frage stellt, werden sowohl die direkt verknüpften Texte, die mit den bisher besuchten Seiten verknüpften Texte und die thematische Ordnung durchsucht. Als Ergebnis wird der Text ausgegeben, der am besten zu den Mustern der Frage passt.

Patrick et al. (1993) untersuchten, ob Dialoge in COMODA inhaltlich aufeinander aufbauen. Sie konnten feststellen, dass die von den Benutzer gestellten Fragen inhaltlich auf die vorherige Antwort aufbauen und Formulierungen aus der Antwort für die neue Frage verwendet werden. Es wurde aber auch festgestellt, dass die Genauigkeit der Antwort des Computers Einfluss auf die Tendenz hat Bezug auf die letzte Antwort zu nehmen.

COMODA beantwortet über 60% der Fragen richtig und wird von über 90% der Benutzer positiv bewertet, wie Patrick und Whalen berichten.

Aus meiner Sicht trifft für COMODA das Argument von Krause (1988) besonders gut zu, dass die Chance für natürlichsprachliche Systeme bei einfachen Problemstellungen gering sind

und die Grenze durch die (inzwischen sehr verbreiteten) Möglichkeiten der direkten Manipulation noch weiter zu Ungunsten der natürlichsprachlichen Interfaces verschiebt.

COMODA ist eine übersichtliche Domäne. So sind in der AIDS-Datenbank 195 Antworten gespeichert. Bei einer guten Organisation der Daten ist es für den Benutzer bei dieser Menge sicherlich noch möglich, mit den Methoden der direkten Manipulation schnell die gesuchte Antwort zu finden. Die Möglichkeiten von COMODA können wohl eher ergänzend gesehen werden, wie man heutzutage in einem Internetangebot auch eine Suchmaschine für die Stichwortsuche finden kann. Nur ist die Aufarbeitung eines Datenbestandes für die Benutzung mit COMODA sehr aufwendig und kann nur manuell gemacht werden[26]. Wenn man sich aber die Mühe macht, die Daten zu erschließen, ist eine Suche mit COMODA sicherlich einer Stichwortsuche überlegen. Bei Antworten, die inhaltlich weiter von der aktuellen Information entfernt sind und nicht durch direkte Verknüpfungen von der aktuellen Seite erreicht werden, können zeigen sich auch Vorteile der COMODA-Technik.

Patrick und Whalen (1992) müssen auch aus ihrer Feldstudie berichten, dass 56% der Versuchspersonen nicht die Möglichkeit zum Fragen genutzt haben, sondern die alternative Möglichkeit, sich mit der Return-Taste die Informationen in einer von den Autoren vorgegebene Reihenfolge anzuschauen, bevorzugten.

Die Anwendungsgebiet von COMODA ist in den letzten Jahren von HTML eingenommen worden. Und ich halte es für vorteilhafter, dass der Benutzer sieht, welche Informationen mit dem gerade gezeigten Thema verbunden und verfügbar sind. Es ist hier ja nicht der Fall, dass zusätzliche Informationen verfügbar werden wenn man danach fragt. Wenn die Notwendigkeit zu fragen besteht können Informationen, die gespeichert sind, nicht gefunden werden, wenn einem nicht klar wird, das man nach ihnen fragen kann.

Menüs oder im Text eingebettete Menüs, wie in HTML, schaffen eine bessere Übersicht über die Domäne, wenn diese vernünftig gestaltet werden. Es ist bei diesen Methoden auch nicht notwendig zu erkennen, was ein „natürliches“ Thema ist.

3.6 DICOS

DICOS ist eine Studie zum Verhalten von Benutzern an natürlichsprachlichen Schnittstellen. Es wurde kein Programm entwickelt, das natürliche Sprache verstehen kann. Statt dessen werden die Äußerungen des Benutzers von einem Menschen beantwortet. Je nach Szenario ist der Versuchsperson dieses auch bewusst.

26 Obwohl es heute sicherlich Methoden gibt, die eine weitgehend automatisierte Aufarbeitung der Daten ermöglichen.

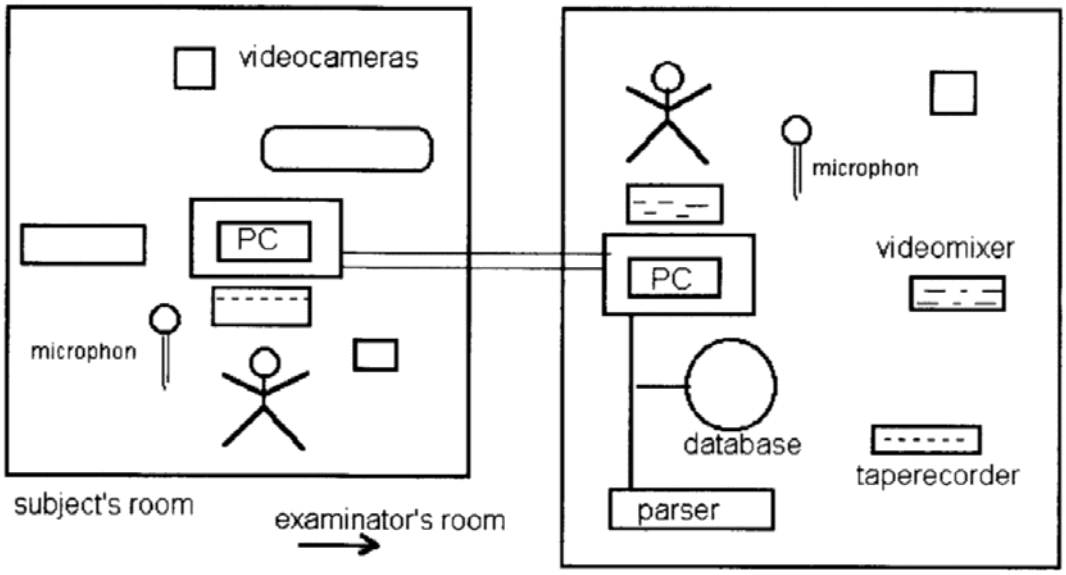


Abbildung 3.1 Aufbau des DICOS Versuches (aus Krause, 1995)

In allen Szenarien sollten die Benutzer in einer Bibliotheksdatenbank und in einen Eisenbahninformationssystem Informationen suchen.

Alle Szenarien werden von unterschiedlichen Versuchspersonen in gesprochener und geschriebener Sprache durchgeführt, so dass auch Unterschiede zwischen den Äußerungen bei den verschiedenen Eingabemethoden gefunden werden können.

Das Hauptinteresse bestand aber darin zu sehen, wie sich Benutzer an die sprachlichen Fähigkeiten der Maschine anpassen und ob die Sprache im Mensch-Maschine-Dialog eine andere ist als zwischen Menschen, auch wenn die Kommunikation in beiden Fällen über einen Computer stattfindet.

Szenario 1: Dem Benutzer wurde gesagt, dass er mit einem Eisenbahn- bzw. Bibliotheksangestellten redet, der in einer Datenbank nachschaut. Der Computer ist nur der Ein- und Ausgabekanal. Sprachliche Äußerungen und die Kooperationsbereitschaft des Systems sind nicht beschränkt. (Krause, 1995)

Szenario 2 ist exakt wie das erste Szenario, nur wurde dem Benutzer gesagt, er kommuniziere mit einem Computer. Er wurde ausdrücklich darauf hingewiesen, dass der Computer natürliche Sprache ohne Einschränkungen versteht und dass er fragen kann was er will. Der einzige Unterschied besteht in dem metalen Konzept des Systems im Kopf des Benutzers. (Krause, 1995)

Szenario 3: Die Kooperationsbereitschaft des Systems ist eingeschränkt. Die Anfragen des Benutzers müssen gewisse Einschränkungen erfüllen, z.B. keine ungenauen oder modalen Ausdrücke, wörtliche Interpretation von Zeitphrasen, usw. Wenn diese Bedingung nicht erfüllt wurde, wird die Eingabe zurückgewiesen und mitgeteilt, weshalb ein Fehler auftrat. (Krause, 1995)

Szenario 4 hat noch einige weitere Einschränkungen und es gibt auch keine Analyse, weshalb ein Fehler aufgetreten ist. Ansonsten verhält es sich aber wie Szenario 3. (Krause, 1995)

Ein Problem dieses Ansatzes ist, dass es einem Mensch, der versucht ein Computerprogramm zu simulieren, nicht möglich ist, in jeder Situation die Restriktionen des Programmes zu erkennen, dessen Verhalten er erzeugen soll. Es entstehen Fehler, so dass das Verhalten des Programmes auf den Benutzer nicht konstant wirken mag und sein Verhalten beeinflusst.

Trotzdem wurden in diese Studie wertvolle Erkenntnisse gesammelt, die bei der Gestaltung einer effizienten natürlichsprachlichen Schnittstelle von großem Wert sind.

Die mir vorliegende Auswertung der DICOS-Studie von Krause (1995) geht vor allem darauf ein, wie häufig bestimmte linguistische Phänomene in den einzelnen Szenarien verwendet werden.

DICOS soll hier nur beispielhaft für eine Vielzahl anderer „Wizard of Oz“ Experimente stehen, in denen einem Benutzer eine natürlichsprachliche Schnittstelle vorgetäuscht wird. Im Rahmen dieser Arbeit haben noch die Experimente von Hauptmann und Rudnicky (1988) und von Zoltan et al. (1982) eine Rolle gespielt.

Hauptmann und Rudnicky haben ein Emailprogramm mit einer natürlichsprachlichen Schnittstelle vorgetäuscht. Es gab drei Szenarien: in einem sagte die Versuchsperson einer anderen Person, was sie wollte, woraufhin diese es am Computer ausführte, im Nächsten sprach es die Versuchsperson in ein Mikrofon, und ihr wurde gesagt, ein sprachverstehendes System würde die Eingabe verarbeiten und als letztes Szenario musste die Versuchsperson die Befehle in natürlicher Sprache eintippen. In Ihrer Auswertung vergleichen Hauptmann und Rudnicky die Äußerungen der einzelnen Szenarien (verwendete Worte, Satzlänge, benötige Äußerungen bis zur Lösung der Aufgabe, benötigte Zeit für die Bearbeitung der Aufgaben, usw.) und gehen auch auf die Fehler ein, die gemacht wurden.

Zoltan et al. benutzen eine Kontoverwaltung als Domäne. Hier gab es zwei Szenarien, in einem wurde getippt und im anderen gesprochen. In der Auswertung beschränken sich Zoltan et al. auf den Vergleich der Äußerungen.

Die im Rahmen dieser Arbeit interessantesten Erkenntnisse aus DICOS und den anderen beiden Experimenten wurden im Kapitel 2.5 dargestellt.

3.7 Schlussfolgerungen

Die Verarbeitung natürlicher Sprache ist in den meisten hier vorgestellten Systemen wenig eindrucksvoll. Nur das USL kann hier wirklich überzeugen, und es wäre sicherlich interessant gewesen zu sehen, wozu NL-Schnittstellen heute fähig wären, wenn dieses System konsequent weiterentwickelt worden wäre. Leider scheint es, dass modernere natürlichsprachliche Systeme einfache Methoden zur Analyse der Eingabe bevorzugen, was dazu führt, dass wir von einer natürlichen sprachlichen Interaktion mit dem Computer wieder weiter entfernt sind.

Der Ansatz, natürlichsprachliche Systeme zu entwickeln, die Syntax und Semantik korrekt auswerten, ist Anfang der '90iger mit dem USL verschwunden. Moderne System nutzen lieber Techniken, die einfacher zu implementieren sind und dabei auch noch schneller und auf weniger leistungsfähigen Rechner arbeiten. Auch wenn die Rechenleistung bei einer Anfrage vielleicht noch nicht von Bedeutung ist, würde aber eine Internetsuchmaschine wie AskJeeves deutlich weniger Anfragen auf der selben Hardware bearbeiten können, wenn eine syntaktische und semantische Analyse der Eingabe vorgenommen würde. Somit ist der Rückgang ambitionierter natürlichsprachlicher Systeme wahrscheinlich zum Großteil auf wirtschaftliche Interessen zurückzuführen, denn es scheint nicht so , dass die Entwicklung damals aufgrund von unlösbaren Problemen bei komplexeren Systemen eingestellt wurde.

Hingegen stellte Krause schon 1988 fest, dass die Chancen für natürlichsprachliche Systeme bei einfachen Problemstellungen sehr gering sind und sich durch die Einführung der Methoden zur direkten Manipulation noch weiter zu Ungunsten der NL-Systeme verschieben.

Betrachtet man die Dialogstrategien der Systeme, kann man diese in 2 Gruppen unterteilen. Zum einen die Mehrzahl der hier vorgestellten Systeme, die vom Benutzer befragt werden und als Reaktion ein Ergebnis liefern[27]. Und zum anderen Systeme, die dem Benutzer so lange befragen, bis ein Ergebnis gefunden wurde. Als Beispiel für diese Art von Programmen ist hier nur das C@ll + Find System vorhanden, aber es gibt eine Vielzahl anderer, vorwiegend im

27 Ask Jeeves zähle ich auch zu diese Gruppe, da das Ausgeben einer Frage als erste Antwort des Systems mit einer Paraphrasierung zu vergleichen ist, wie sie das USL vornimmt.

Bereich telefonischer Systeme; außerdem auch die schon einmal erwähnten Assistenten, wie den Internet-Verbindungsassistenten und ähnliche.

Die Idee, eine Datenbank durch wechselseitige aufeinander aufbauende Dialoge abzufragen, wird noch von keinem mir bekannten System genutzt. Die meisten Systeme vernachlässigen Dialoge, andere benutzen zwar Dialoge, sehen diese aber nur als eine Umsetzung von Formularen, wodurch Fragen ausgewählt werden, die ausschließlich minimale Antworten erzeugen, wodurch viele andere Vorteile von Dialogen vernachlässigt werden. Es ist somit festzustellen, dass keines der hier vorgestellten Systeme die Möglichkeiten von Dialogen wirklich ausnutzt.

Ein Dialog ermöglicht es, dass der Benutzer die Struktur einer Datenbank nicht von vornherein kennen muss; die beiden klassischen Datenbanksysteme Q&A und USL, bei denen dies von größtem Nutzen wäre, nutzen dies nicht.

Im weiteren kann auch keines der System gefragt werden, welche Daten es anbietet, z.B. „Kannst du etwas über die Qualität der Restaurants sagen?“.

Ein Dialog sollte Abkürzungen zulassen und es erlauben, dass der Benutzer auch Daten nicht angeben muss. Keines der System nutzt ein solches Dialogdesign, im Q&A sind sogar zusätzliche Dialogschritte notwendig, um eine komplexe Anfrage bearbeiten zu können. In anderen Systemen wie C@ll+Find ist der Dialogverlauf starr vorgegeben und die Fragen lassen keine Abkürzungen zu.

Ein Dialogsystem kann die notwendigen logischen Verknüpfungen zwischen zwei Aussagen automatisch erstellen und somit vor dem Benutzer verbergen. C@ll+Find nutzt dies zwangsläufig, die Relationen sind hier aber fest vorgegeben und müssen nicht vom System aus der Syntax und dem Kontext bestimmt werden, die anderen Systeme nutzen diese Möglichkeit überhaupt nicht.

Bei keinem der hier vorgestellten Systeme findet ein Wechsel der Initiative statt – entweder fragt nur der Computer oder nur der Benutzer.

Es ist in den Systemen, die den Benutzer befragen, nicht möglich eine Frage nicht zu beantworten und statt dessen eine andere möglicherweise benötigte Information zu geben.

Kein System nutzt die Möglichkeit mit einem Dialog seine Einschränkungen aufzuzeigen und den Benutzer darauf hinzuweisen wie weit die Funktionalität des Systems reicht und was außerhalb der Möglichkeiten des Systems liegt, z.B. „Das Wort 'brauche' ist unbekannt, bitte verwenden sie ein anderes Wort!“

4 Beschreibung des implementierten Systems „Telefonauskunft“

Da viele der Ansätze für dialoggesteuerte Datenbankabfrage noch in keinem System implementiert wurden, habe ich im Rahmen dieser Arbeit eine Telefonauskunft programmiert, in der ich versucht habe, einen Teil der Ideen zu realisieren, die sich durch den Dialogansatz ergeben haben.

Bei dem Programm handelt es sich um eine Auskunft für Adressen und Telefonnummern von Personen. In einem weiteren Schritt wurde es um die Möglichkeit nach Firmen zu suchen erweitert. Die aktuelle Version erlaubt das Suchen nach Firmen über der Firmennamen und die Gewerbebezeichnung.

Die Initiative im Dialog kann in diesem Programm wechseln. Die erste Anfrage an das System kann der Benutzer frei formulieren. An dieser Stelle kann er auch schon Abkürzungen nutzen, indem er z.B. sowohl die gesuchte Person als auch den Ort angibt. Das System wird daraufhin beginnen, durch Nachfragen noch fehlende Informationen vom Benutzer zu erhalten. Der Benutzer hat aber jederzeit die Möglichkeit, die Befragung abzubrechen und z.B. die Ausgabe der Ergebnisse zu verlangen. Falls der Benutzer den Dialog nicht abbricht, wird das System versuchen, die Anzahl der Ergebnisse so weit wie möglich durch Nachfragen zu reduzieren.

Der Benutzer braucht die Struktur der Datenbank nicht kennen, im Dialog wird auf leicht verständliche Art und Weise nach den Werten für die Spalten gefragt. Wenn dem Benutzer, z.B. durch öfteres Benutzen des Systems, bekannt ist, welche Angaben das System benötigt steht es ihm natürlich frei, den Dialog abzukürzen und die benötigten Informationen in einer Aussage anzugeben.

Fragen brauchen vom Benutzer nicht direkt beantwortet zu werden, wenn die Antwort nicht nur Stichworte enthält, sondern eine klare Aussage steht es dem Benutzer frei, z.B. eine Frage nach dem Ort mit der Straße, in der die gesuchte Person lebt zu beantworten.

Die einzelnen Dialogzüge des Benutzers werden vom Programm automatisch verknüpft, so dass die Schnittmenge der Aussagen gebildet wird. Es steht dem Benutzer frei innerhalb einer Aussage selber semantisch gleiche Begriffe mit „oder“ oder „und“ zu verknüpfen, wobei hier in beiden Fällen davon ausgegangen wird, dass die Vereinigungsmenge gebildet werden soll (siehe auch Kapitel 2.3.5).

Methoden, die es dem System erlauben, Fragen zu seiner Funktionalität oder den Daten, die es anbietet, zu beantworten gibt, wurden nicht implementiert. Die Fehleranalyse in diesem Programm ist nicht so weit fortgeschritten. Unbekannte Worte können vom System

identifiziert werden, aber die Analyse der Eingabe geht nicht so weit, dass es dem System daraufhin möglich ist, gemeinsam mit dem Benutzer eine Lösung zu erarbeiten. Die Dialogkomponente ist dazu ebenfalls nicht imstande.

4.1 Vorhandene Funktionalität

Es handelt sich bei dem Programm um eine Telefonauskunft bei der man nach Personen und Firmen suchen kann. Der Benutzer interagiert mit dem System in einem natürlichsprachlichen Dialog. Eine detailiertere Beschreibung der Eigenschaften bieten die folgenden Abschnitte.

4.1.1 Der Client

Das Programm funktioniert als Client für eine SQL-Datenbank, als Server wird nur die Datenbank benötigt. Es können mehrere Clienten gleichzeitig gestartet werden, wenn die IP-Adresse der Datenbank im Programm eingetragen wird kann der Client auch auf unterschiedlichen Rechnern laufen, wenn eine Netzwerkverbindung besteht.

4.1.2 Die Grammatik

Das Programm analysiert nur Hauptsätze und einfache W-Fragen. Kompliziertere Satzkonstruktionen können möglicherweise verarbeitet werden, da der Parser recht flexibel ist. Die Qualität der Ergebnisse leidet dann möglicherweise unter den Einschränkungen, wie sie auch für nicht grammatikalisch richtige Sätze gelten.

Die Grammatik ist auf geschriebene Sprache ausgelegt. Wie in Kapitel 2.5.2 gibt es Unterschiede in der Syntax bei geschriebener und gesprochener Sprache.

Die Grammatik ist sicherlich nicht optimal codiert und es bedarf noch einiger Verbesserungen, aber sie weist dafür eine gewisse Toleranz bei Sätzen auf, die grammatikalisch nicht ganz korrekt sind, da falsche Eingaben dieser Art nicht auszuschließen sind.

Es werden viele syntaktische Varianten unterstützt, um eine Aussage zu machen. Das heißt, für eine Aussage sind verschiedene Schreibweisen möglich.

Der Benutzer kann frei formulierte Anfrage an das System stellen. Frei formulierbar natürlich nur insoweit, wie das System in der Lage ist die Grammatik und das Vokabular der

Anfrage zu verstehen. Wie schon erwähnt, können natürlichsprachliche Systeme nur eine Teilmenge der natürlichen Sprache verarbeiten. Die Teilmenge in diesem Programm ist relativ klein.

4.1.3 Das Lexikon

Die Verben sind nur in ihrer Präsensform gespeichert, sie können getrennt werden (z.B. Gib ... aus), und auch passive Satzkonstruktionen sind möglich.

Das Programm kennt ca. 140 Vokabeln; die meisten sind Worte, andere bestehen aus Phrasen. Wo es nötig war, sind die Vokabeln auch als Vollformen gespeichert. In den meisten Fällen sind aber nur die benötigten Schreibweisen eingetragen.

4.1.4 Systemverhalten

Die Begrüßung durch das System ist bewusst so gewählt, dass dem Benutzer keine Formulierung für seine Anfrage vorgegeben ist. Ziel dabei ist es, zu erfahren, wie Benutzer eine Anfrage an eine Telefonauskunft stellen würde, wenn er frei entscheiden kann.

Das System bezeichnet sich im Ausgabefenster selbst als Auskunft. Der Begriff ist neutral und weist nicht sofort darauf hin, dass der Benutzer mit einer Maschine kommuniziert.

Bei einigen Nachfragen stehen mehrere Fassungen derselben Frage zur Verfügung, um beim Benutzer den Eindruck von vorgefertigten Standardfragen zu vermeiden. Ob dies sinnvoll ist, muss noch überprüft werden, bei mehrmaligem Benutzen des Systems dürfte der Benutzer aber schnell alle Varianten entdecken. Wo es inhaltlich möglich ist, werden auch noch Daten wie der Name oder das Geschlecht der gesuchten Person eingefügt. Die Fragen sind so formuliert, dass die erste Person vermieden wird. Der Computer soll nicht personalisiert werden, um die Vorteile des Computertalk für die Verarbeitung der Sprache nutzen zu können. In dieser Erwartung sind auch keine Höflichkeitsfloskeln kodiert worden.

Vor jeder Nachfrage des Systems wird ausgegeben, wie viele Ergebnisse bisher gefunden wurden. Dem Benutzer soll damit transparenter gemacht werden, weshalb eine Frage gestellt wird.

4.1.5 Suchoptionen

Eine Anfrage muss nicht den Namen der Person oder Firma enthalten. Es ist auch möglich zu sagen „Ich suche einen Anschluss in Hamburg“. Die Dialogkomponente wird dann aber höchstwahrscheinlich irgendwann nach dem Namen der gesuchten Person oder Firma fragen.

Felder mit der selben Semantik (z.B. Straßen) können mit „oder“ und „und“ kombiniert eingegeben werden. Bei der Datenbankabfrage werden alle so kombinierten Worte als mögliche Werte benutzt.

Wenn eine Anfrage alle benötigten Informationen enthält, werden keine Nachfragen gestellt, sondern das Ergebnis sofort ausgegeben.

Auch wenn das Programm darauf ausgelegt ist, natürliche Sprache zu verarbeiten, kann man nicht immer damit rechnen, dass der Benutzer auch Sätze eingibt. Das Programm funktioniert auch wenn nur Stichworte eingegeben werden, z.B. nur der Name. Anweisungen müssen auch nicht als vollständiger Satz eingegeben werden, es reicht zum Beispiel „ausgeben“ zu tippen.

Eine Firma braucht nicht über ihren Namen gesucht zu werden, alternativ kann man das Gewerbe der Firma angeben. Dahinter steckt die Idee, dass Benutzer bei einer Firma oft genug der Name der Firma egal ist, sondern nur die Dienstleistung oder die Waren, die diese Firma anbietet, interessieren. Dass dem Nutzer eine Zuordnung der gesuchten Ware oder Dienstleistung zu einer Gewerbebezeichnung möglich ist, ist aus meiner Sicht zu erwarten.

4.1.6 Steuerbarkeit

Der Benutzer kann die Nachfragen des Systems abbrechen und eine Ausgabe aller bisher gefundenen Ergebnisse verlangen (z.B. durch einen Satz wie „Zeige mit alle Ergebnisse“) oder alternativ auch einzeln durch die Ergebnisse gehen, mit Sätzen wie „Zeige mir die erste Adresse“ und „Gib den nächsten Eintrag aus“. Falls man sich die Ergebnisse einzeln ausgeben lässt, kann man sich auch mit „Zeig mir den Rest“ alle Ergebnisse ab dem aktuellen bis zum letzen ausgeben lassen.

Auch bei der initialen Anfrage kann man die Ausgabe der gesamten Ergebnismenge verlangen, indem man zum Beispiel „Zeig mir alle Meyer aus Osnabrück“ eintippt.

Das Ausgabeformat des Suchergebnisses kann vom Benutzer angegeben. Er kann sagen, dass er nur die Telefonnummer oder die Adresse (Name, Straße und Ort) oder die komplette Adresse (Name, Straße , Ort und Telefonnummer) sehen will. Wenn nichts angegeben ist, wird die komplette Adresse ausgegeben.

Wenn ein Ergebnis oder eine Fehlermeldung ausgegeben wurde, kann der Benutzer eine neue Suche beginnen. Es ist aber auch möglich, das Ergebnis noch weiter einzuschränken oder das Ausgabeformat zu ändern. Eine andere Gelegenheit für einen Intentionswechsel außer in dieser Situation gibt es im Programm nicht.

Das Programm kann auch natürlichsprachlich beendet werden indem der Benutzer sich vom Programm verabschiedet (z.B. Auf Wiedersehen).

4.1.7 Die Dialogkomponente

Die Dialogkomponente ist sehr flexibel. Es werden nur Fragen gestellt, die notwendig sind, um ein Ergebnis zu finden, d.h. wenn Informationen schon vom Benutzer genannt wurden oder aus den bisher gefundenen Daten erkennbar ist, dass eine Information nicht benötigt wird, dann wird die jeweilige Nachfrage auch nicht gestellt. Die Auswahl der zu stellenden Frage erfolgt immer nach dem Kriterium, welche Frage die Ergebnismenge am meisten verringert.

Die Nachfragen sollen es dem Benutzer ermöglichen, die Struktur der Datenbank nicht kennen zu müssen. Ein Grundlegendes Wissen über die Domäne Telefonauskunft wird aber schon vom Benutzer erwartet, da der erste Dialogzug von ihm ausgeführt wird.

Es wird vom Benutzer nicht erwartet, dass er alle Fragen des Systems beantwortet. Wie im vorigen Abschnitt erwähnt, kann der Benutzer wieder die Initiative im Dialog übernehmen und z.B. die Ausgabe der Ergebnisse verlangen oder eine andere Information angeben.

Der Dialog erlaubt es dem Benutzer, mehrere Aussagen in der Suche miteinander zu verketten, ohne dass er sich darüber Gedanken machen muss, ob diese nun mit „und“ oder „oder“ verknüpft sein müssen.

4.1.8 Fehleranalyse

Es gibt zwei Arten von Fehlermeldungen. Die eine Fehlermeldung tritt auf, wenn das Parsen der Eingabe nicht einwandfrei war, und die andere tritt auf, falls keine Ergebnisse gefunden wurden. In beiden Fällen werden, wenn möglich, Informationen präsentiert, die zu dem Fehler geführt haben. Bei Parserfehlern wird angegeben, welches Wort nicht erkannt wurde, und

welche Informationen trotzdem aus der Eingabe gewonnen wurde. Der Benutzer muss dann angeben, ob das Angezeigte dem entspricht, was er eingeben wollte, wenn dem nicht so ist, muss er die Eingabe umformulieren.

Wenn kein Ergebnis gefunden wurde, werden die eingegebenen Daten angezeigt, die nicht zueinander passen, also z.B. es gibt keinen Schröder, der an einer Straße Waldweg wohnt oder es gibt am Waldweg keine Hausnummer 26. Leider sind die Korrekturmöglichkeiten im Programm noch nicht so weit entwickelt, dass es möglich ist, den angezeigten Fehler auch zu beseitigen.

4.1.9 Fehlertoleranz

Das Programm weist, außer bei Eigennamen von Personen und Firmen, auch keine Toleranz bei Rechtschreibfehlern auf.

Straßennamen können als solche erkannt werden, auch wenn sie nicht in der Datenbank gespeichert sind und zwar in dem Fall wenn sie für Straßen übliche Endungen besitzen, wie z.B. Straße, Gasse, Weg und Platz. Die Fehlermeldung kann durch diese Analyse präzisiert werden.

4.2 Architektur

Das Programm hat eine einfache Benutzerschnittstelle, hauptsächlich ein kleines Textfeld für die Eingabe des Benutzers und ein größeres Textfeld in dem die Ausgabe des Programmes und auch die bisherigen Eingaben des Benutzers stehen.

Die Eingabe des Benutzers wird an einen Parser weitergeleitet, der diese analysiert und die für das Programm wesentlichen Daten extrahiert. Mit diesen Daten wird die Datenbank abgefragt.

Die Datensätze aus dieser Abfrage werden analysiert und je nach dem Ergebnis der Analyse wird der Text für eine Nachfrage, Fehlermeldung oder Ausgabe der Datensätze generiert.

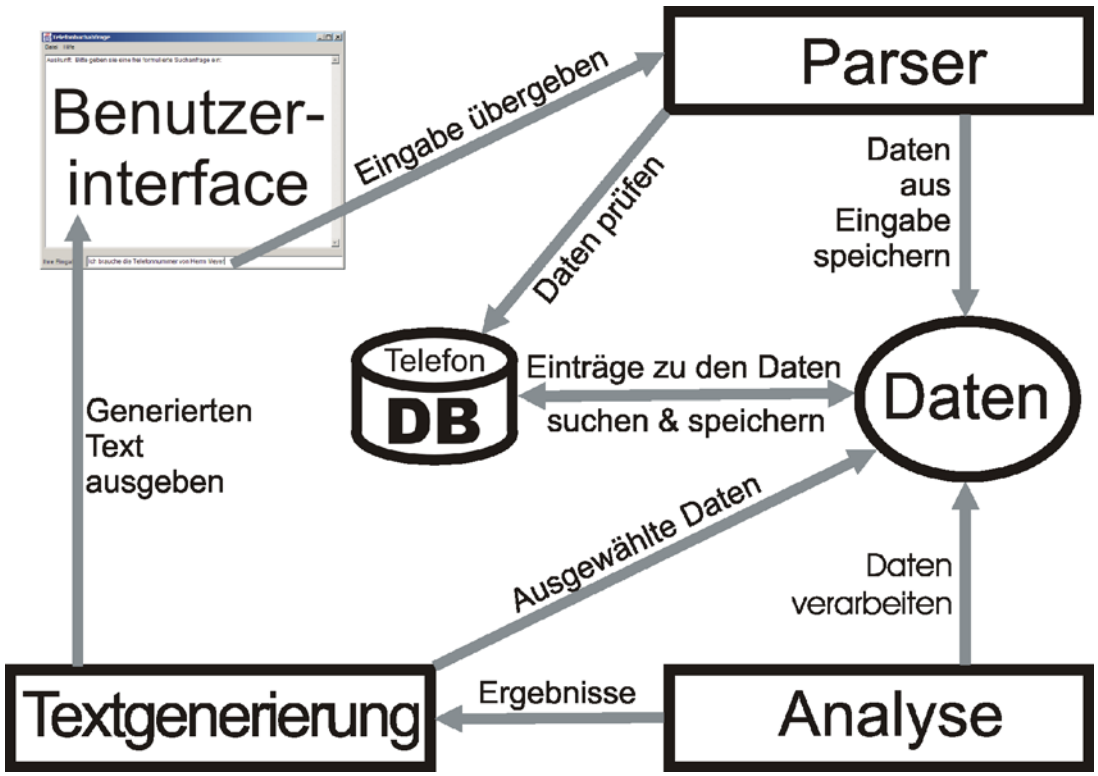


Abbildung 4.1 Skizze des Systems

4.2.1 Das Benutzerinterface

Das Benutzerinterface ist, wie schon gesagt einfach, gehalten. Der Benutzer soll angeregt werden, natürliche Sprache einzusetzten.

Der Aufbau des Interface ist ähnlich wie bei einem Chat-Programm. In einem großen Textfeld wird die Eingabe des Benutzers und die Ausgabe des Systems protokolliert, und in einem kleinen Textfeld darunter kann der Benutzer seine Eingabe machen.

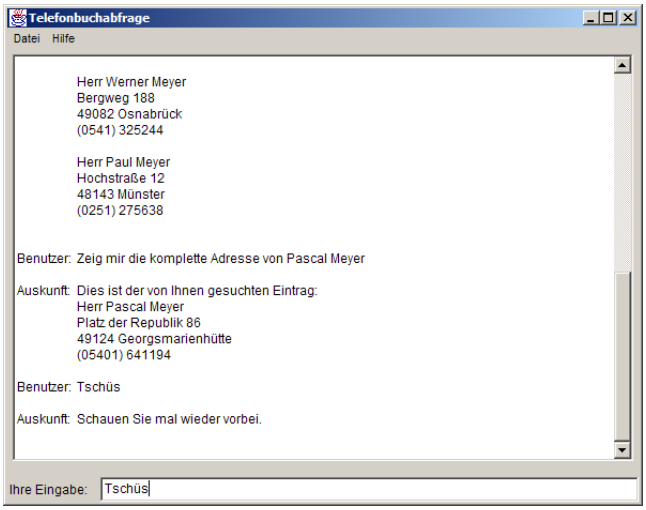


Abbildung 4.2 Benutzerinterface mit Buttons

Je nach Programmversion befinden sich unter dem Eingabefeld noch einige Knöpfe für häufige Antworten („Ja“, „Nein“, „Das weiß ich nicht“, Zeig mir alle Ergebnisse“). Diese zweite Programmversion wurde für eine mögliche vergleichende Evaluierung erstellt. In der innerhalb dieser Arbeit vorgestellten Evaluation wurde nur die in Abbildung 4.2 gezeigte Oberfläche verwendet.

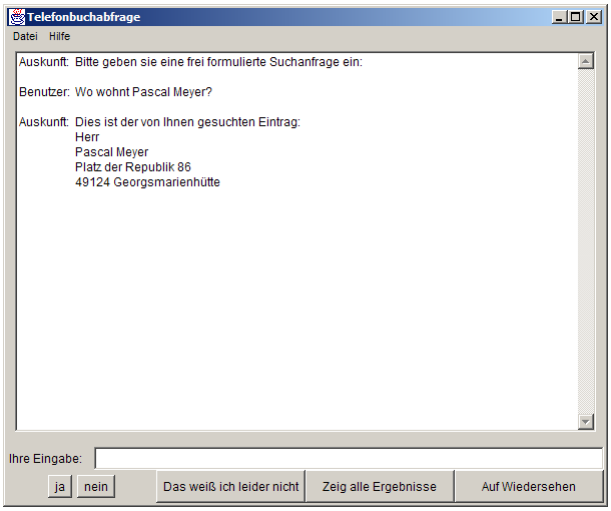


Abbildung 4.3 Das Benutzerinterface mit Knöpfen für häufige Eingaben

Es gibt natürlich noch ein Menü für die wichtigsten Programmfunktionen: „Hilfe“, „Neue Anfrage“ und „Programm beenden“. Wobei „Neue Anfrage“ nur für den Fall gedacht ist, dass der Benutzer den Dialog abbrechen will um komplett neu anzufangen. Es ist im Dialog natürlich möglich mehrere Anfragen hintereinander zu stellen.

Der Knopf „Neue Anfrage“ ist auch noch zusätzlich in einem freischwebenden Menü vorhanden (nicht auf den Abbildungen zu sehen).

Die Protokollierung in der Ausgabe ist als Hilfestellung für den Benutzer gedacht, so dass er z.B. sehen kann, welche Informationen er schon eingegeben hat oder ob er bei der Eingabe einen Fehler gemacht hat.

In einem anwendungsorientierten Programm kann es später sinnvoll sein, mehre Befehle und Funktionen als in dem in Abbildung 4.3 gezeigten Prototypen auf Buttons oder in Menüs vorzugeben. Hier aber war es mir wichtig, dass das Programm sich über die Sprache steuern läßt, auch wenn dies bedeutet, dass die Bedienbarkeit darunter leidet.

4.2.2 Der Parser

Der wesentlich Teil des Parsers ist in Prolog geschrieben. Der Prologcode besteht hauptsächlich aus DCG-Klauseln.

Ein externes Wörterbuch wird nicht benutzt. Das benötigte lexikalische Wissen ist als Vollformen im Prologcode enthalten.

Namen, Straßen und Orte werden anhand ihrer Präpositionen, der Stellung im Satz und ihrem Kontext vom Prologprogramm erkannt. Eine Überprüfung, worum es sich bei dem gefundenen Wort wirklich handelt, erfolgt erst im Javaprogramm. Dort wird die Datenbank abgefragt, ob es einen passenden Eintrag zu dem vermuteten Worttyp gibt. Bei Personennamen werden ähnliche Namen, nach der „Kölner Phonetik“ (Postel, 1969) mit einbezogen. Bei Straßennamen wird zusätzlich kontrolliert, ob ein Bestandteil des Ausdrucks (Straße, Gasse, Weg, usw.) darauf schließen läßt, dass es sich hierbei um einen Straße handelt.

Tritt bei der Kontrolle in Java ein Fehler auf, werden über das Prologprogramm die Alternativen gesucht. Wenn keine ganz korrekte Alternative gefunden wird, wird das erste Ergebnis noch einmal genauer untersucht und alle verwertbaren Informationen gespeichert. Außerdem wird dann für die Textgenerierung festgehalten, dass beim Parsen ein Fehler aufgetreten ist. Aus Performancegründen ist die Berechnung auf 100 Alternativen beschränkt worden.

Um Fehler zu erkennen wird für jeden gefunden Eintrag ein Plausibilitätscheck durchgeführt. Wenn alle 100 Alternativen kein Ergebnis liefern wird ein weiterer Test des Wortes über alle möglichen Werte für dieses Wort gemacht. Nur Gewerbebezeichnungen sind noch nicht in diesem Test erfasst.

Die aus der Eingabe erhaltenen Informationen werden natürlich auch in einem Datenobjekt gespeichert, wenn kein Fehler aufgetreten ist.

Neben Personen- und Firmennamen, Straßen und Orten erfasst der Parser auch noch andere Informationen:

- Der Parser kann erkennen, ob eine Person oder Firma gesucht wird.
- Wenn nach einer Person gesucht wird, wird das Geschlecht erfasst.
- Wenn nach einer Firma gesucht wird, kann man auch das Gewerbe angeben, in dem diese Firma tätig ist. Die Gewerbe sind als Vollformen im Prologcode eingetragen (Die Erstellung der Gewerbe erfolgt im einem Generator-Tool, mit dem auch die Datenbankeinträge erzeugt werden können. Die Gewerbe können also auch von einem Benutzer ohne Programmierkenntnisse ergänzt werden).

- Das Ausgabeformat der Daten kann vom Benutzer angegeben werden. Er kann z.B. sagen, das er nur die Telefonnummer der gesuchten Person haben möchte.
- Befehle an das System werden erkannt, wie die Ausgabe aller noch vorhanden Ergebnisse oder das Beenden des Programmes.
- Der Benutzer kann sich iterativ die Ergebnisse anzeigen lassen („Ich möchte gerne das nächste Ergebnis“)
- Natürlich werden auch Postleitzahlen oder Hausnummern erkannt.
- Oder-Verknüpfte Ausdrücke können erkannt werden, z.B.: „Er wohnt in Osnabrück oder Münster.“
- Bei Zahlenwerten sind auch Reihen erlaubt (Waldweg 12 bis 29).

Der berücksichtigte Kontext besteht aus der Frage, die dem Benutzer gestellt wurde, beziehungsweise den Startzustand und den Präpositionen und Artikeln die dem unbekannten Wort vorausgehen.

Im Folgenden ein Ausschnitt aus dem in Prolog programmierten Teil des Parsers. Die Klauseln wurden etwas bereinigt um die Übersichtlichkeit zu erhöhen. Es fehlt aber lediglich die in der restlichen Implementation nicht weiter unterstützte Funktionalität mit der später möglicherweise Referenzen aufgelöst werden können. Die hier gezeigten Ausschnitte sind der Teil des Parsers der Namen erkennt (siehe auch Kapitel 2.2.2.1):

Kurze Erklärung eines „normalen“ Prädikats. Die Werte in eckigen Klammern werden je nach Wortart gebraucht:

- Pradikatsname ([Kasus],[Nummerus],[Genus],[Person],[aktiv/passiv], Syntax der letzten Frage, Semantik dieses Prädikats, bisherige Ergebnisse, Ergebnisse dieser Klausel + bisherige Ergebnisse) .

Ausschnitt aus der Datei „td-de.pl“:

```
                                        ...
auswerten(LetzteFrage,Ergebnis) -->
      frage(LetzteFrage,[],Ergebnis),
      frageende.
auswerten(LetzteFrage,Ergebnis) -->
      hs(LetzteFrage,[],Ergebnis),
      satzende.
```

Was hier in der Variablen „Ergebnis“ steht wird vom Java-Programm weiterverarbeitet.

```
                                        ...

frage(LF, E1,[(ansicht,adresse)|E]) -->
      fragewort,
```

```
      vp(sg,_,3,aktiv,LF,wohnen,E1,E2),
      np(nom,_,_,_, LF,name,E2,E).
frage(LF,E1,[(ansicht,adresse)|E]) -->
      fragewort,
      vp(pl,_,3,aktiv,LF,wohnen,E1,E2),
      aufzaehlung(nom,_,_,_, LF,name,E2,E).
frage(LF,E1,[(ansicht,adresse)|E]) -->
      fragewort,
      vp(sg,_,3,aktiv,LF,wohnen,E1,E2),
      np(nom,_,_,_, LF,name,E2,E3),
      aufzaehlung(_,_,_,_, LF,_,E3,E).
frage(LF,E1,[(ansicht,adresse)|E]) -->
      fragewort,
      vp(pl,_,3,aktiv,LF,wohnen,E1,E2),
      aufzaehlung(nom,_,_,_, LF,name,E2,E3),
      aufzaehlung(_,_,_,_, LF,_,E3,E).

                                        ...

hs(LF,E1,E) -->
      np(nom,Numerus,Genus,Agens, LF,ich,E1,E2),
      vp(Numerus,Genus,1,Agens, LF,suchen,E2,E3),
      np(akk,_,_,_, LF,name,E3,E).

                                        ...
hs(LF,E1,E) -->
      np(nom,Numerus,Genus,Agens, LF,ich,E1,E2),
      vp(Numerus,Genus,1,Agens, LF,suchen,E2,E3),
      np(gen,_,_,_, LF,name,E3,E4),
      np(akk,_,_,_, LF,ansicht,E4,E).

                                        ...

hs(LF,E1,E) -->
      np(nom,Numerus,Genus,Agens, LF,ich,E1,E2),
      vp(Numerus,Genus,1,Agens, LF,suchen,E2,E3),
      np(akk,_,_,_, LF,name,E3,E4),
      np(dat,_,_,_, LF,strasse,E4,E5),
      np(dat,_,_,_, LF,ort,E5,E).

                                        ...
```

Ausschnitt aus der Datei „td-de-np.pl“:

```
np(Kasus,Numerus,Genus,Agens,LF,S,E1,E) -->
      n(Kasus,Numerus,Genus,Agens,LF,S,E1,E).
np(Kasus,Numerus,Genus,Agens,LF,S,E1,E) -->
      partikeln(E1,E2),
      n(Kasus,Numerus,Genus,Agens,LF,S,E2,E).
np(Kasus,Numerus,Genus,Agens,LF,S,E1,E) -->
      n(Kasus,Numerus,Genus,Agens,LF,S,E1,E2),
      partikeln(E2,E).
np(Kasus,Numerus,Genus,Agens,LF,S,E1,E) -->
      n(Kasus,Numerus,Genus,Agens,LF,S,E1,E2),
      komma,
      partikeln(E2,E).
np(Kasus,Numerus,Genus,_,_,ich,E1,E) -->
      nppronomen(Kasus,Numerus,Genus,1,E1,E).
np(Kasus,Numerus,Genus,_,_,pronomen,E1,E) -->
      nppronomen(Kasus,Numerus,Genus,1,E1,E).
np(Kasus,Numerus,Genus,Agens,LF,S,E1,E) -->
      pronomen(Kasus,Numerus,Genus,_,E1,E2),
      np(Kasus,Numerus,Genus,Agens,LF,S,E2,E).
```

...

```
% Namen
n(Kasus,Numerus,Genus,_, _,gewerbe,E1,[(gesucht,firma),(gewerbe,ID)|E1]) --
>
      pp(Kasus,firma),  gewerbe(Kasus,Numerus,Genus,ID).
n(Kasus,Numerus,Genus,_, _,gewerbe,E1,[(gesucht,firma),(gewerbe,ID)|E1]) --
>
      artikel(Kasus,Numerus,Genus,ndet), gewerbe(Kasus,Numerus,Genus,ID).
n(Kasus,Numerus,Genus,_, _,gewerbe,E1,[(gesucht,firma),(gewerbe,ID)|E1]) --
>
      gewerbe(Kasus,Numerus,Genus,ID).
n(Kasus,Numerus,Genus,_, _,name,E1,[(gesucht,firma),(gewerbe,ID)|E1]) -->
      artikel(Kasus,Numerus,Genus,ndet),
      gewerbe(Kasus,Numerus,Genus,ID).
n(Kasus,sg,Genus,_, _,name,E1,E) -->
      pp(Kasus,firma), firma(Kasus,Genus,E1,E2),
      name(Kasus,Genus,firmenname,E2,E).
n(Kasus,sg,Genus,_, _,name,E1,E) -->
      pp(Kasus,firma),
      artikel(Kasus,sg,Genus,det),
      name(Kasus,Genus,firmenname,E1,E).
n(Kasus,sg,Genus,_, _,name,E1,E) -->
      pp(Kasus,firma), name(Kasus,Genus,firmenname,E1,E).
n(Kasus,sg,Genus,_, _,name,E1,E) -->
      firma(Kasus,Genus,E1,E2),
       name(Kasus,Genus,firmenname,E2,E).
n(Kasus,sg,Genus,_, _,name,E1,E) -->
      pp(Kasus,person),
      geschlecht(Kasus,Genus,E1,E2),
      name(Kasus,Genus,_,E2,E).
n(Kasus,sg,m,_, _,name,E1,[(geschlecht,m)|E]) -->
      pp(Kasus,person),
      artikel(Kasus,sg,m,_),
      name(Kasus,m,name,E1,E).
n(Kasus,sg,w,_, _,name,E1,[(geschlecht,w)|E]) -->
      pp(Kasus,person),
      artikel(Kasus,sg,w,_),
      name(Kasus,w,name,E1,E).
n(Kasus,sg,Genus,_, _,name,E1,E) -->
      geschlecht(Kasus,Genus,E1,E2),
      name(Kasus,Genus,_,E2,E).
n(Kasus,sg,n,_, _,name,E1,[(gesucht,firma)|E]) -->
      artikel(Kasus,sg,m,_),
      name(Kasus,m,name,E1,E).
n(Kasus,sg,m,_, _,name,E1,[(geschlecht,m)|E]) -->
      artikel(Kasus,sg,m,_),
      name(Kasus,m,name,E1,E).
n(Kasus,sg,w,_, _,name,E1,[(geschlecht,w)|E]) -->
      artikel(Kasus,sg,w,_),
      name(Kasus,w,name,E1,E).
n(Kasus,sg,Genus,aktiv,vname,vname,E1,E) -->
      name(Kasus,Genus,vorname,E1,E).
n(Kasus,sg,Genus,aktiv,name,vname,E1,E) -->
      name(Kasus,Genus,vorname,E1,E).
n(Kasus,sg,Genus,aktiv,nname,nname,E1,E) -->
      name(Kasus,Genus,nachname,E1,E).
n(Kasus,sg,Genus,aktiv,name,nname,E1,E) -->
      name(Kasus,Genus,nachname,E1,E).
n(Kasus,sg,Genus,aktiv,name,name,E1,E) -->
      name(Kasus,Genus,nachname,E1,E).
n(Kasus,sg,Genus,aktiv,name,name,E1,E) -->
      name(Kasus,Genus,name,E1,E).
```

```
n(Kasus,sg,Genus,aktiv,start,name,E1,E) -->
      name(Kasus,Genus,nachname,E1,E).
n(Kasus,sg,Genus,aktiv,start,name,E1,E) -->
      name(Kasus,Genus,name,E1,E).
n(Kasus,sg,Genus,aktiv, _,name,E1,E) -->
      pp(Kasus,person),
      name(Kasus,Genus,_,E1,E).
n(Kasus,sg,Genus,aktiv, _,name,E1,E) -->
      name(Kasus,Genus,_,E1,E).
```

...

Die komplette Datei „td-de-name.pl“:

```
% Geschlecht der Person bestimmen
geschlecht(Kasus,Genus,E1,[(geschlecht,Genus),(gesucht,person)|E1]) -->
      geschlechtsworte(Genus,Kasus).
geschlecht(Kasus,Genus,E1,[(geschlecht,Genus),(gesucht,person)|E1]) -->
      artikel(Kasus,sg,Genus,_),
      geschlechtsworte(Genus,Kasus).


% Geschlechtsworte
geschlechtsworte(w,_) --> [frau].
geschlechtsworte(w,_) --> [fräulein].

geschlechtsworte(m,nom) --> [herr].
geschlechtsworte(m,gen) --> [herrn].
geschlechtsworte(m,dat) --> [herrn].
geschlechtsworte(m,akk) --> [herrn].

% Indikatoren für eine Firma
firma(Kasus,Genus,E1,[(gesucht,firma)|E1]) -->
      firma(Genus,Kasus).
firma(Kasus,Genus,E1,[(gesucht,firma),(gewerbe_id,ID)|E1]) -->
      gewerbe(Kasus,sg,Genus,ID).
firma(Kasus,Genus,E1,[(gesucht,firma)|E1]) -->
      artikel(Kasus,sg,Genus,_), firma(Genus,Kasus).
firma(Kasus,Genus,E1,[(gesucht,firma),(gewerbe,ID)|E1]) -->
      artikel(Kasus,Numerus,Genus,_ ),
      gewerbe(Kasus,Numerus,Genus,ID).

% Bezeichnungen für Firma
firma(f,_) --> [firma].
firma(m,_) --> [betrieb].
firma(n,_) --> [geschaeft].

% Namen erkennen
name(Kasus,_,name,E1,E) -->
      nachname(Kasus,E1,E2),
      komma,
      vorname(Kasus,E2,E).
% obere Klausel wird nicht vom JPL berücksichtigt!!!!!!!!!!!!
name(Kasus,_,name,E1,E) -->
       [namens],
      name(Kasus,E1,E).
name(Kasus,_,nachname,E1,E) -->
       [namens],
      nachname(Kasus,E1,E).
name(Kasus,_,name,E1,E) -->
       [mit,dem,namen],
      name(Kasus,E1,E).
name(Kasus,_,nachname,E1,E) -->
```

```
      [mit,dem,namen],
      nachname(Kasus,E1,E).
name(Kasus,_,nachname,E1,E) --> nachname(Kasus,E1,E).
name(Kasus,_,firmenname,E1,E) --> firmenname(Kasus,E1,E).
name(Kasus,_,vorname,E1,E) --> vorname(Kasus,E1,E).
name(Kasus,_,name,E1,E) -->
      [namens],
      vorname(Kasus,E1,E2),
      nachname(Kasus,E2,E).
name(Kasus,_,name,E1,E) -->
      [mit,dem,namen],
      vorname(Kasus,E1,E2),
      nachname(Kasus,E2,E).
name(Kasus,_,name,E1,E) -->
      vorname(Kasus,E1,E2),
      nachname(Kasus,E2,E).

nachname(Kasus,E1,[(nachname,[Nachname,zu,Nachname2],Kasus)|E1]) -->
      [Nachname, zu, Nachname2].
nachname(Kasus,E1,[(nachname,[Nachname, - ,Nachname2],Kasus)|E1]) -->
      [Nachname], minus, [Nachname2].
nachname(Kasus,E1,[(nachname,[von,der, Nachname],Kasus)|E1]) -->
      [von,der, Nachname].
nachname(Kasus,E1,[(nachname,[von, Nachname],Kasus)|E1]) -->
      [von, Nachname].
nachname(Kasus,E1,[(nachname,[van, Nachname],Kasus)|E1]) -->
      [van, Nachname].
nachname(Kasus,E1,[(nachname,Nachname,Kasus)|E1]) -->
      [Nachname].

name(Kasus,E1,[(name,Name,Kasus)|E1]) --> [Name].

vorname(Kasus, E1,[(vorname,[Vorname,-,Vorname2],Kasus)|E1]) -->
      [Vorname],
      minus,
      [Vorname2].
vorname(Kasus,E1,[(vorname,Vorname,Kasus)|E1]) --> [Vorname].

firmenname(Kasus,E1,[(firma,[Name,-,Name2],Kasus)|E1]) -->
      [Name],
      minus,
      [Name2].
firmenname(Kasus,E1,[(firma,[Name,&,Name2],Kasus)|E1]) -->
      [Name],
      [xxund],
      [Name2].
firmenname(Kasus,E1,[(firma,Name,Kasus)|E1]) -->
      [Name].
firmenname(Kasus,E1,[(firma,[Name,Name2],Kasus)|E1]) -->
      [Name],
      [Name2].
```

4.2.3 Das Datenobjekt

Das Datenobjekt speichert alle vom Parser gesammelten Informationen, um mit diesen Daten die Datenbank abzufragen. Es werden auch alle Ergebnisse der Anfrage in diesem Objekt zur Weiterverarbeitung gespeichert. Aus den bekannten Informationen werden in

diesem Objekt, wenn möglich, fehlende Daten ergänzt (z.B. aus dem Vornamen das Geschlecht oder aus der Postleitzahl den Ort bestimmen).

Die Methoden, um die Datenbank abzufragen, gehören auch zu diesem Objekt und die Ergebnisse der Abfrage werden in diesem Objekt gespeichert.

Alle für eine Suche wichtigen Daten sind in diesem Objekt gespeichert, so dass, wenn eine neue Suche gestartet werden soll, man nur das Datenobjekt neu initialisieren muss. Das Programm merkt sich aber das alte Datenobjekt. In einer späteren Programmversion kann versucht werden aus dem alten Objekt Daten für die aktuelle Suche zu übernehmen.

4.2.4 Analyse der Datenbankabfrageergebnisse

Um zu ermitteln, wie die nächste Frage des Programmes aussieht, falls nicht das Ergebnis oder eine Fehlermeldung ausgegeben werden sollen, muss das Ergebnis der Datenbankabfrage analysiert werden.

Dazu wird gezählt, wie viele unterschiedliche Werte die einzelnen Felder enthalten. Das Feld mit den meisten unterschiedlichen Werten wird für die nächste Frage vorgemerkt.

Falls mehrere Felder gleich gut für eine Nachfrage geeignet sind, wird nach einer manuell erstellten Wertung die Frage ausgewählt, die dem Benutzer hoffentlich am sinnvollsten erscheint. Die Reihenfolge, in der die Fragen bei identischer Bewertung ausgewählt werden, ist:

1. Ort
2. Nachname oder Firmennamen
3. Straße
4. Vorname oder Gewerbebezeichnung

Dies ist nur ein Vorschlag, ob diese Reihenfolge dem entspricht, was der Benutzer erwartet, kann nur durch eine Evaluation geklärt werden. In einer Evaluation kann auch geklärt werden, ob der unbedingte Vorrang der Auszählung vor der erwarteten Reihenfolge der Fragen vom Benutzer nachvollzogen werden kann.

4.2.5 Generieren der Systemreaktion

Es gibt einen Entscheidungsbaum (siehe Abbildung 4.4), in dem ausgewählt wird, welche Systemreaktion gewählt wird. Wenn die Entscheidung, welche Reaktion ausgeführt werden soll, getroffen worden ist, wird die Textgenerierung gestartet. Im Anschluss wird gespeichert, welche Frage gestellt wurde. Danach wird der vom System erzeugte Text ausgegeben.

Die folgenden Reaktionen sind möglich:

- Ein Ergebnis wird ausgegeben, wenn die Anzahl der Ergebnisse unter eine voreingestellte Schwelle gefallen ist. In der aktuellen Programmversion ist diese Schwelle genau Eins, dies kann aber leicht geändert werden. Ein weiterer Schwellwert kann eingestellt werden, ab dem eine Nachfrage gestellt wird, ob alle verbleibenden Ergebnisse ausgegeben werden sollen.
- Wenn es kein Ergebnis gibt, werden wenn möglich die Gründe ausgegeben, weshalb Nichts gefunden wurde, in manchen Fällen scheitert aber die Analyse, weshalb die Suche scheiterte.
- Falls ein Fehler beim Parsen aufgetreten ist, wird eine Rückfrage gestellt, um die Informationen, die trotz des Fehlers gewonnen wurden, bestätigen zu lassen oder, falls die gewonnen Daten falsch sind eine Korrektur der letzten Eingabe zu ermöglichen.

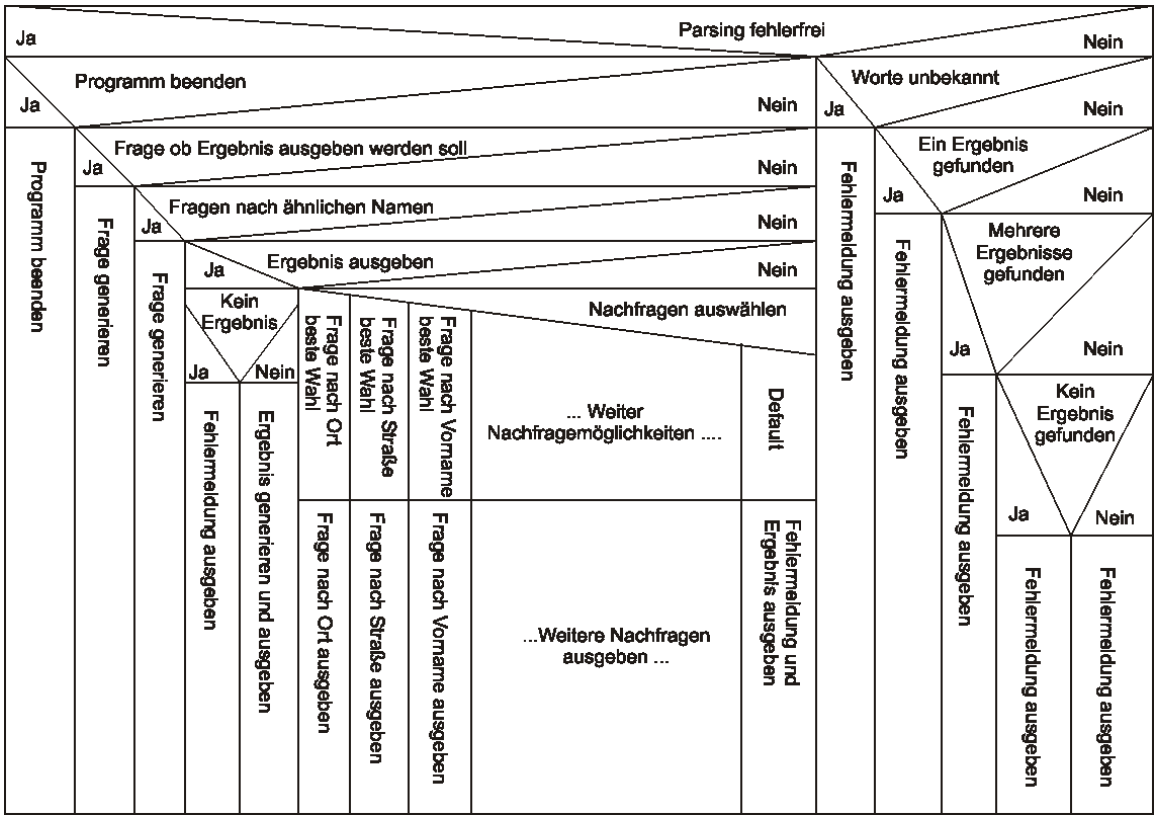


Abbildung 4.4 Leicht vereinfachter Entscheidungsbaum

4.3 Aufbau der Datenbank

Es wird in diesem Programm eine SQL-Datenbankverwendet, die aus 10 Tabellen besteht, von den aber nur 4 momentan vom Programm verwendet werden. Auf die teilweise schon vorbereiteten oder geplanten Erweiterungen wird im Kapitel 4.6 eingegangen.

Die Tabelle Personen enthält die Personen mit Adressen, nach denen gesucht werden kann. P_ID ist eine eindeutige Bezeichnung für jeden Eintrag, die automatisch vergeben wird. In NName ist der Nachname und in VName der Vorname der Person gespeichert. NNphon und

VNphon sind die Einträge für die phonetische Repräsentation nach der Kölner Phonetik (Postel, 1969). Auf den Ort wird durch die Ort_ID verwiesen.

Atribut	***Datentyp***	***Nullwerte***	***Schlüssel***	***Defaultwert***	***auto_increment***
P_ID	int(11) unsigned	Ja[28]	Ja	NULL	Ja
NName	mediumtext	Ja	Nein		
NNphon	mediumtext	Ja	Nein		
VName	mediumtext	Ja	Nein		
VNphon	mediumtext	Ja	Nein		
Strasse	text	Ja	Nein		
HNr	tinyint(4) unsigned	Ja	Nein	0	
Ort_ID	int(11) unsigned	Ja	Nein	0	
Telefonnr	int(10) unsigned	Ja	Nein	0	
Geschlecht	tinytext	Ja	Nein		

Tabelle 4.1 Die Tabelle Personen in der Telefondatenbank

Die Tabelle Firmen enthält alle gespeicherten Firmen. F_ID ist eine eindeutige Bezeichnung für jeden Eintrag in dieser Tabelle, der Wert wird automatisch zugewiesen. FNphon ist die phonetische Repräsentation des Firmennamens, der unter FName gespeichert ist. Auf den Ort wird durch eine Ort_ID in der Tabelle Orte verwiesen.

Atribut	***Datentyp***	***Nullwerte***	***Schlüssel***	***Defaultwert***	***auto_increment***
F_ID	int(11) unsigned	Ja	Ja	NULL	Ja
FName	mediumtext	Ja	Nein		
FNphon	mediumtext	Ja	Nein		
Strasse	text	Ja	Nein		
HNr	tinyint(4) unsigned	Ja	Nein	0	
Ort_ID	int(11) unsigned	Ja	Nein	0	
Telefonnr	int(10) unsigned	Ja	Nein	0	
Gewerbe_ID	int(11) unsigned	Ja	Nein		

Tabelle 4.2 Die Tabelle Firmen in der Telefondatenbank

28 Kann nicht wirklich NULL sein, da der Wert automatisch vergeben wird, wenn NULL zugewiesen wird.

Ein Designfehler ist beim Anlegen der Datenbank passiert, so dass mehrere Vornamen, Doppelnamen und Firmennamen die aus mehreren Worten bestehen in nur einem Feld beim jeweiligen Eintrag gespeichert werden, so dass nicht nach den einzelnen Teilen des Namens gesucht werden kann. Wenn man also einen „Jens-Uwe Meier“ sucht ist es nicht möglich, nur „Jens Meier“ oder gar „Uwe Meyer“ zu suchen, um den Eintrag zu finden. Eine kleinere Nachbesserung habe ich vorgenommen, die phonetische Suche kann auch nur auf dem ersten Namensteil suchen. Wenn ein „Jens Meier“ nicht bekannt ist wird man gefragt, ob der Vorname vielleicht nur ähnlich ist. Wenn man dies bejaht, wird der passende Eintrag auch gefunden.

Die Tabelle Orte enthält auch eine eindeutige Ort_ID. Der Ortsname, die Postleitzahl (PLZ) und Vorwahl sind in dieser Tabelle gespeichert. Bei der Vorwahl ist die vorangehende 0 (bzw. +49 für Deutschland) nicht mitgespeichert, da ein int-Wert zum Speichern in der Datenbank verwendet wird. Die 0 muss von der Anwendung ergänzt werden. Auch bei Postleitzahlen, die kleiner als 1000 sind, ist die vorangehende 0 von der Anwendung zu ergänzen.

Atribut	***Datentyp***	***Nullwerte***	***Schlüssel***	***Defaultwert***	***auto_increment***
Ort_ID	int(11) unsigned	Ja	Ja	NULL	Ja
Ortsname	mediumtext	Ja	Nein		
PLZ	mediumint(5) unsigned	Ja	Nein	0	
Vorwahl	mediumint(5) unsigned	Ja	Nein	0	

Tabelle 4.3 Die Tabelle Orte in der Telefondatenbank

Die Tabelle Gewerbe enthält eine Gewerbe_ID, eine allgemeine Bezeichnung für das Gewerbe, den Genus des Wortes und alle für die Flexion notwendigen Fälle. Die Gewerbe_ID ist nicht mehr eindeutig, mehrere Worte die ein Gewerbe bezeichnen erhalten eine ID.

Atribut	***Datentyp***	***Nullwerte***	***Schlüssel***	***Defaultwert***	***auto_increment***
Gewerbe_ID	int (11) unsigned	Nein	Nein		
Bezeichnung	mediumtext	Ja	Nein		
Genus	tinytext	Ja	Nein		
Genitiv_sg	mediumtext	Ja	Nein		
Dativ_sg	mediumtext	Ja	Nein		

Plural	mediumtext	Ja	Nein		
Dativ_pl	mediumtext	Ja	Nein		
Akkusativ_pl	mediumtext	Ja	Nein		

Tabelle 4.4 Die Tabelle Gewerbe aus der Telefondatenbank

Im weiteren noch kurz die unbenutzten Tabellen und ihre angedachte Funktion:

- Tabelle Abteilungen: Eine Firma kann aus mehreren Abteilungen bestehen. Eine Abteilung kann eine eigene Adresse und Telefonnummer haben. Eine Abteilung hat zusätzlich zu einem Namen auch noch eine Funktion, die in der Tabelle Funktionen vordefiniert ist. Jede Abteilung besitzt eine eigene eindeutige A_ID (Abteilungs-ID).
- Tabelle Angestellte: In dieser Tabelle werden Personen mit Abteilungen über die IDs verknüpft. Zusätzlich gibt es noch Spalten für den Beruf, den Raum und die Telefonnummer des Angestellten.
- Tabelle Berufe: Alle dem System bekannten Berufe. Falls die Tabelle benutzt werden soll, wird sie wahrscheinlich noch so geändert werden müssen, dass sie so ähnlich wie die Tabelle Gewerbe aussieht. Bisher werden die Berufe noch nicht als Vollformen gespeichert.
- Tabelle Berufsgruppen: Durch diese Tabelle sollen ähnlich Berufe zusammengefasst werden, damit auch nach Personen bzw. Angestellten mit ähnliche Berufen gesucht werden können.
- Tabelle Funktionen: Eine Liste der Funktionen, die eine Abteilung haben kann. Auch hier muss die Tabelle noch so geändert werden, dass die Funktionen als Vollformen gespeichert werden.
- Tabelle Gewerbegruppen: Wie bei der Tabelle Berufe sollen hier ähnliche Gewerbe zusammengefasst werden können, um die Möglichkeit zu haben nach ähnlichen Gewerben zu suchen. Diese Funktion ist bisher so realisiert, dass mehrere Gewerbe unter einer Gewerbe_ID zusammengefasst werden (s.o.).

4.4 Ablauf des Programmes

Um eine Vorstellung darüber zu geben, wie das Programm funktioniert, möchte ich an dieser Stelle kurz ein ein paar beispielhafte Suchanfragen zeigen:

Beispiel 1:

```
Auskunft:   Bitte geben sie eine frei formulierte Suchanfrage ein:
Benutzer:   Ich suche Meyer
Auskunft:   8 Personen gefunden. Damit die Menge kleiner wird müssen
            noch ein paar Fragen gestellt werden.
            In welcher Strasse wohnt Meyer?
```

```
Benutzer:   Die kenne ich nicht
Auskunft:   8 Einträge entsprechen Ihrer Anfrage. Wenn Sie deshalb
            noch eben folgende Fragen beantworten:
            Wie ist denn der Vorname von Meyer?
Benutzer:   Pascal
Auskunft:   Dies ist der von Ihnen gesuchten Eintrag:
            Herr Pascal Meyer
            Platz der Republik 86
            49124 Georgsmarienhütte
            (05401) 641194
```

Beispiel 2:

```
Benutzer:   Ich suche Lenin aus Berlin
Auskunft:   Leider wurde kein Eintrag gefunden.
            Eine Person mit dem Nachnamen Lenin gibt es nicht in
            13403 Berlin .
```

Beispiel 3:

```
Benutzer:   Ich brauche die Telefonnummer von Herrn Ralf
Auskunft:   Ist der Nachname so richtig geschrieben?
Benutzer:   nein#
Auskunft:   Ihre Eingabe konnte nicht korrekt verarbeitet werden.
            Leider kein Ergebnis gefunden.
            Folgende Informationen konnten Ihrer letzten Eingabe
            entnommen werden:
            ein Nachname "nein#" ist in dieser oder ähnlicher
            Schreibweise nicht bekannt.
            Korregieren Sie bitte ihre Angaben!
Benutzer:   nein
Auskunft:   Ist der Nachname so richtig geschrieben?
Benutzer:   nein
Auskunft:   Dies ist der von Ihnen gesuchten Eintrag:
            Rüdiger Rolf: (0541) 59466
```

Beispiel 4:

```
Benutzer:   Ich suche Herrn Meyer
Auskunft:   4 Einträge entsprechen Ihrer Anfrage.
            Deshalb werden Ihnen noch folgenden Fragen gestellt:
            Und in welcher Strasse wohnt Herr Meyer?
Benutzer:   Er wohnt in Onabrück
Auskunft:   Ihre Eingabe konnte nicht korrekt verarbeitet werden. Es
            wurden trotzdem 4 Ergebnisse gefunden.
            Folgende Informationen konnten Ihrer letzten Eingabe
            entnommen werden:
            ein Ort namens "Onabrück" ist nicht bekannt.
            Möchten sie die letzte Eingabe korrigieren?
Benutzer:   ja
Auskunft:   4 Personen gefunden. Deshalb werden Ihnen noch folgenden
            Fragen gestellt:
            Und in welcher Strasse wohnt Herr Meyer?
Benutzer:   Er wohnt in Osnabrück
Auskunft:   Dies ist der von Ihnen gesuchten Eintrag:
            Herr Werner Meyer
            Bergweg 188
            49082 Osnabrück
            (0541) 325244
```

Beispiel 5:

```
Benutzer:   Ich suche eine Bank
Auskunft:   Es gibt noch 2 Firmen, die Ihrer Anfrage entsprechen. Um
            das Ergebnis weiter einzuschränken beantworten Sie bitte
```

```
            die folgenden Fragen.
            In welchem Ort soll die Bank sein?
Benutzer:   Osnabrück
Auskunft:   Hier sind alle auf Ihre Anfrage gefundenen Einträge:
            Bank
            Sparkasse
            Kuhweg 101
            49082 Osnabrück
            (0541) 723258
```

4.5 Protokollierung

Alles was in dem Ausgabefeld an Text erscheint wird protokolliert. Zusätzlich zum Text wird noch die Uhrzeit, wann der Text in die Ausgabe geschrieben wurde, sekundengenau erfasst. Beim Start des Programmes und wenn „Neue Anfrage“ aus dem Menü gewählt wurde, wird ein Kopf mit dem Datum und der Uhrzeit geschrieben. Die Protokolle werden im Arbeitsverzeichnis des Programmes unter „protokol-ndd.txt“ und „protokol-nddbuttons.txt“ je nach Benutzerinterface gespeichert.

Bei jedem Programmstart wird eine neue Protokoll-ID vergeben. Bei jedem Programmstart und jedesmal wenn „Neue Anfrage“ aus dem Datei-Menü gewählt wird wird eine neue Anfrage-ID vergeben. Um die IDs zurückzusetzen, muss die Datei „protokoll.id“ im Verzeichnis des Programms gelöscht werden.

4.6 Verbesserungen und Erweiterungen für das Programm

In unterschiedlichen Bereichen sind Änderungen am Programm mehr oder weniger leicht möglich. Manche sind nur auf Zeitgründen nicht weiter verfolgt worden, andere Konzepte sind in der Realisierung wahrscheinlich viel aufwendiger als das gesamte hier vorgestellte Programm. Im folgenden soll ein Überblick über diese Änderungen gegeben werden.

4.6.1 Erweiterung und Verbesserung des Datenbestandes

Von Beginn an ist das Programm auf eine Datenbank mit mehr Tabellen konstruiert worden. Es sollte noch die Möglichkeit geben, nach Abteilungen einer Firma und Angestellten suchen zu können. Das Programm ist in vielen Teilen auch schon darauf vorbereitet (siehe „Aufbau der Datenbank“). So enthält das Datenobjekt die Felder für die zusätzlich benötigten Daten. Die Klasse, die zum Abfragen der Datenbank vorhanden ist, enthält die Methoden, um nach Angestellten und Abteilungen zu suchen und neue Daten von dieser Art in die Datenbank einzutragen. Der Generator (siehe „zusätzliche Programme“) kann diese Daten in die

Datenbank eintragen. Und natürlich sind in der Datenbank auch die benötigten Tabellen vorhanden.

Eine Firma kann nach diesem Konzept beliebig viele Abteilungen haben. Die Abteilungen besitzen eine eigene Adresse und Telefonnummer, einen Abteilungsnamen und eine Funktion der Abteilung innerhalb der Firma. Der Abteilungsname ist normalerweise identisch mit dem Funktionsnamen, kann aber vom Benutzer beliebig vergeben werden (z.B. kann die Abteilung mit der Funktion „Verkauf“ auch „Export“ heißen). Die Funktionen sind nur aus der Tabelle Funktionen auswählbar.

Ein Angestellter ist eine Person, die einer Abteilung zugeordnet ist. Ein Angestellter besitzt einen Beruf, eine Telefonnummer und einen Raum. Die Adresse ist somit zwangläufig identisch mit der Adresse der Abteilung. Die Berufe sind aus der Tabelle Berufe auswählbar. Jeder Person kann nur ein Beruf zugeordnet werden.

Personen sollen auch über ihren Beruf zu suchen sein, wie es bei Firmen schon über das Gewerbe möglich ist. Natürlich sollte eine Person auch über ihren Arbeitgeber bzw. die Abteilung, in der sie arbeitet, zu finden sein.

Es ist angedacht, dass ähnliche Berufe und ähnliche Gewerbe über die Tabellen Berufsgruppen bzw. Gewerbegruppen zusammengefasst werden können. Unter einer Gruppen-ID können mehrere einzelne IDs der Berufe oder Gewerbe zusammengefasst werden. Diese Funktionalität dient dazu, Alternativen zu finden, falls eine direkte Suchen nach dem eingegeben Beruf bzw. Gewerbe nicht das passende Ergebnis bringt.

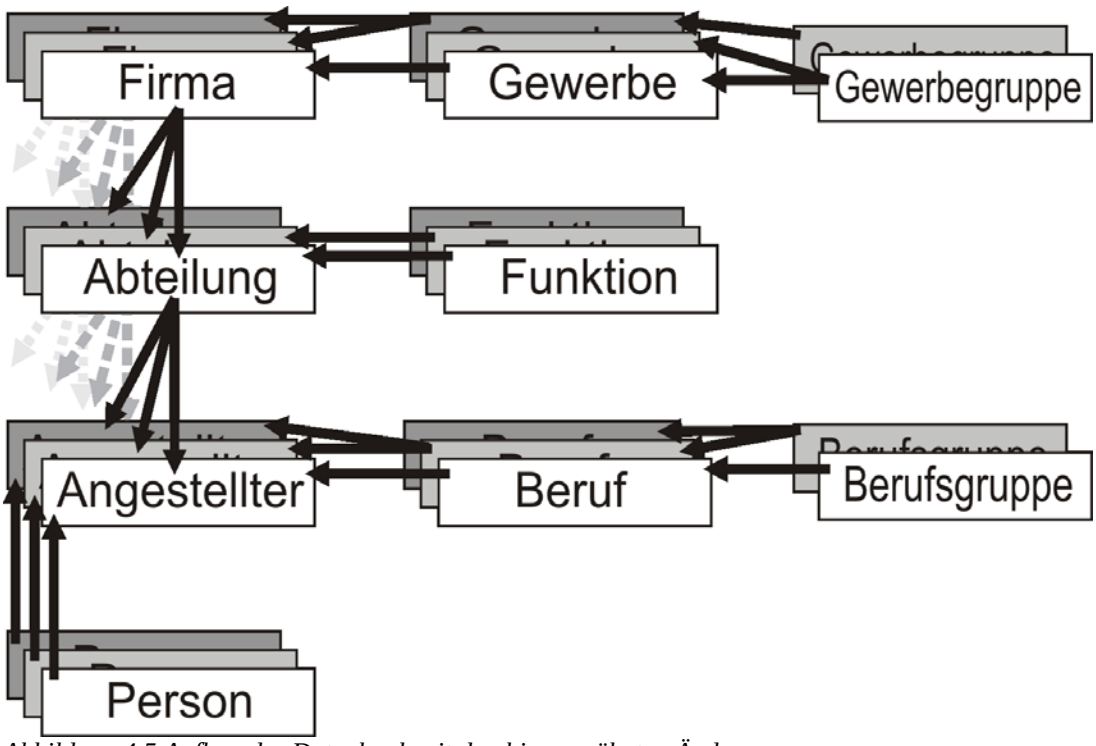


Abbildung 4.5 Aufbau der Datenbank mit den hier erwähnten Änderungen

Wie gesagt ist vieles für die Realisation dieser Erweiterungen schon vorbereitet. Die benötigten Änderung sind vor allem am Parser vorzunehmen, der wohl nicht nur um die benötigten Funktionen erweitert, sondern auch in Teilen effizienter gestaltet werden muss, damit die Änderungen nicht schon vorhandene Funktionen außer Kraft setzen.

In den Tabellen für Personen und Firmen müssen noch Spalten für Doppelnamen, mehrere Vornamen, Firmennamen, die aus mehreren Worten bestehen, geschaffen werden. Auch die Methoden zur Datenbankabfrage müssen so angepasst werden, dass nach einzelnen Teilen dieser Namen gesucht werden kann.

Bei einigen Namen und besonders Firmennamen sind unterschiedliche Schreibweisen möglich (z.B Mediamarkt und Media Markt) dies muss bei der Suche auch noch berücksichtigt werden. Möglich wäre es mehrere Namen zu einem Eintrag zu speichern, oder den Namen automatisch zu zerlegen.

4.6.2 Verbesserung der Korrekturmöglichkeiten

Die Korrekturmöglichkeiten, bei einer falschen Eingabe oder einem Tippfehler des Benutzers, sind momentan sehr beschränkt. Es ist nicht möglich, einen falsch gespeicherten Namen zu ändern. Die benötigte Verbesserung ist, dass Phrasen wie „nicht Kohl sondern Meyer“ erkannt und ausgewertet werden können.

4.6.3 Erweitern der syntaktischen und semantischen Funktionalität

Negationen des Benutzers werden momentan nicht berücksichtigt, da sie meines Erachtens in diesem Kontext nicht sehr häufig vorkommen. Es ist aber durchaus vorstellbar, dass nach einem Herrn Meyer gesucht wird, von dem man nur sicher weiß, dass er nicht Jürgen mit Vornamen heißt, oder einen Klempner außer der Firma Röhrig usw.

Im weiteren kann das Vokabular natürlich beliebig erweitert werden, was die Qualität des Progammes sicherlich steigert. Worte, die in diesem Zusammmenhang besonders sinnvoll wären, sind zum Beispiel „Nachbar“, „Mitbewohner“, „Sohn“, „Kollege“ und so weiter. Natürlich müssen diese Relationen dann auch definiert werden. Neben einzelnen Worten müssten dem System auch noch mehrere Phrasen wie „oder so ähnlich“ bekannt sein.

Das Programm sollte vielleicht irgendwann in der Lage sein, ungewöhnliche sprachliche Konstruktionen zu erlernen. Selbst wenn diese grammatisch falsch sind, kann es sinnvoll sein, diese aufzunehmen, wenn sie häufig vorkommen. (Zoeppritz, 1995:S.50)

Wenn man nicht nur die Telefon-Datenbank zugrunde legt, sondern auch die Straßen in Beziehung zueinander setzt[29] können natürlich auch Beziehungen wie „in der Nähe von ...“, „im Umkreis von 500 Metern zu mir“ oder „das nächste Restaurant“ verarbeiten. Wobei Begriffe wie „Nähe“ immer Relativ sind, „die Kirche ist in der Nähe von meiner Wohnung“ – „Münster ist in der Nähe von Osnabrück“ beides ist eine korrekte Verwendung des Begriffs, die beschriebene Entfernung ist aber sehr unterschiedlich, vielleicht 50 m zu über 50 km. (vgl. Zoeppritz, 1988)

Erweitert man das durch die Straßenkarte gewonnene Wissen auch noch um ein wenig Weltwissen ist es auch möglich, Angaben zu machen, dass die Person „im Norden der Stadt“ wohnt, oder man kann den Ortsteil angeben, falls der in der Karte gespeichert ist.

Legt man in einem nächsten Schritt auch noch andere Datenbanken zugrunde, wie Restaurantführer oder Firmenbewertungen der Verbrauerzentralen, kann man das System auch um eine Beratungskomponente erweitern. Dann sind Anfragen nach einem guten italienischem Restaurant oder einer zuverlässigen Autowerkstatt denkbar.

Auch sollte eine Hilfefunktion implementiert werden, die vom Benutzer natürlichsprachlich aktiviert werden kann, oder sogar am Benutzerverhalten erkennt, wann sie benötigt wird. Diese Hilfe sollte aber, um konsistent zu sein, auch natürlichsprachlich und dialoggesteuert sein und nicht, wie sonst üblich, in einem extra Hilfe-Bildschirm. Es muss sowohl eine

29 Falls man reale Straßen benutzt kann man z.B. eine Navigationssoftware benutzen

kontextabhängige als auch eine allgemeine Hilfe geben, die den Umgang mit dem Programm erklärt.

Referenzen, Anaphern und Ellipsen werden bisher nicht aufgelöst. Ein Ansatz dazu ist zwar im Programm vorhanden, aber wegen des Implementationsaufwands und wegen der Performance des Programmes nicht weiter von mir verfolgt worden. Aus meiner Sicht beeinträchtigt das Fehlen dieser Funktion das Ergebnis der Auswertung in diesem Programm nicht. Dies kann an der Aufgabe der Programms, der Einschränkung auf Hauptsätze und wenige abfragbare Tabellen liegen, oder daran, dass die Referenz maximal nur zur gerade aktuellen Frage hergestellt wird. Vor den Erweiterungen um Berufe und Abteilungen, usw. ist es aber ratsam, eine Möglichkeit zu finden, Anaphern aufzulösen.

Der Parser muss außerdem noch um komplexere Satzkonstruktionen erweitert werden, wie z.B. Relativsätze, Infinitivsätze, Appositionen.

4.6.4 Verbesserung der Datenaufbereitung

Eine sinnvolle Erweiterung ist es auch, dem Benutzer die Möglichkeit zu geben, dass er sagen kann, dass das Ergebnis nach einem beliebigen, vom Benutzer angegeben Kriterium sortiert werden soll.

4.6.5 Verbesserung der Dialogkomponente

Eine Verbesserung für die Dialogkomponente sind allgemeinere Nachfragen, z.B. „Kennen Sie die Adresse von ...?“, die Abkürzungen im Dialog zulassen, bzw. „Wo wohnt ...?“ die flexiblere Antworten ermöglichen, hier Ort und Straße.

Als einen weiteren Schritt benötigen die Fehlermeldungen noch einiger Überarbeitung. Es bedarf für eine gute Fehlermeldung einer Analyse, was zu diesem Fehler geführt hat und welche Informationen bei einem bestimmten Fehler für den Benutzer überhaupt von Bedeutung sind. Momentan wird alles, was bei dem Fehler überhaupt relevant sein könnte, ausgegeben. Die sprachliche Formulierung der Fehlertexte ist wegen der möglicherweise großen Menge von Daten, die an dieser Stelle ausgegeben werden können, auch noch nicht sehr elegant.

Die Auswahl der zu stellenden Frage sollte die Erwartung des Benutzers mit einbeziehen, welche Frage er an dieser Stelle im Programmablauf als sinnvoll betrachtet. Die bisherige Methode kann schätzungsweise zu Verwirrung führen, wenn z.B. nach der Straße gefragt

wird, obwohl man den Ort noch gar nicht angegeben hat. In den Ergebnissen der Evaluation in Kapitel 5 werde ich noch näher darauf eingehen.

Um die Verarbeitung der Eingabe zu verbessern, ist es eventuell nötig, die Fragen so zu formulieren, dass minimale Antworten wahrscheinlicher werden.

4.6.6 Verbesserung der Textgenerierung

Bei der Generierung der Rückfragen sollten mehr schon bekannte Informationen mit eingebunden werden, um diese indirekt bestätigen zu lassen bzw. den Benutzer über Befunde zu informieren, die ihm noch gar nicht bewusst sind, z.B. „In welcher Stadt liegt diese Hauptstraße?“ oder „Einen Herrn Zimmermann gibt es nur in Münster. Können Sie mir vielleicht sagen in welcher Straße er wohnt?“.

4.6.7 Verbesserte Erfassung des Benutzerverhaltens

Intentionswechsel des Benutzers sollten auch zu einem beliebigen Zeitpunkt im Programmablauf möglich sein, wenn der Benutzer diesen deutlich artikuliert oder es an seinem Verhalten deutlich zu erkennen ist.

4.6.8 Benutzermodellierung

Der Benutzer könnte sich gegenüber dem Programm identifizieren (z.B. Einloggen mit einem Benutzernamen, die IP-Adresse seines Computers, Internet-Cookies, Rufnummerübermittlung beim Telefon), so dass vom Programm ein Profil des Benutzers erstellt werden kann.

Der Aufenthaltsort kann benutzt werden, um Relationen wie „in der Nähe“ aufzulösen, bzw. kann gespeichert werden, dass der Aufenthaltsort wechselt, weil der Benutzer z.B. ein Handy benutzt.

Es können ungewöhnliche sprachliche Konstruktionen, die nur von diesem Benutzer verwendet werden, gespeichert werden, damit mit diesem Wissen die Verarbeitung der Eingaben verbessert werden.

Die Strategie des Benutzers bei der Anfrage kann gespeichert werden. Durch Analysen dieser Strategie kann man dem Benutzer Optimierungen seines Verhaltens vorschlagen.

Häufig von einem Benutzer gesuchte Personen oder Firmen können vorgemerkt werden, so dass sie auch gefunden werden können, ohne das alle notwendigen Daten abgefragt wurden.

Aus dem typischen Verhalten des Benutzers können vielleicht Daten gewonnen werden, die für die aktuelle Suche übernommen werden können (z.B. alle bisher gesuchten Telefonnummern kamen aus einem Vorwahlbereich, dann kann man vielleicht annehmen, oder nachfragen, ob die jetzt gesuchte Nummer auch aus diesem Bereich kommen soll).

Aber auch ohne das Speichern eines Benutzerprofils ist es möglich, ein kurzzeitiges Benutzermodell zu erstellen. Aus der vorangegangen Suche können Informationen, wie das Ausgabeformat oder die Stadt, in welcher der gesuchte Teilnehmer leben soll, entnommen werden. Es ist natürlich besser, solche Annahmen nicht stillschweigend zu machen, aber die Dialogkomponente bietet ja die Möglichkeit, sich Informationen vom Benutzer bestätigen zu lassen.

4.6.9 Verbesserung und Erweiterung der Benutzerschnittstelle

An der Benutzerschnittstelle können zum einen die in den Kapitel 2.4.2 (Turn Taking) und 2.4.1 (Ergänzung der Eingabe) gemachten Vorschläge verwirklicht werden. Damit ist gemeint, dass automatisch oder durch eine Aktion des Benutzers (z.B. drücken der Tab-Taste) Worte ergänzt werden, falls sie eindeutig sind und dass das Programm die Eingabe des Benutzers überwacht und reagiert, sobald die Eingabe verstanden wurde.

Praktisch ist auch noch die Funktion, dass man mit den Pfeiltasten die letzten Benutzereingaben wiederholen kann, wie man dies z.B. auch von der Unix-Shell kennt.

Die vorhandene Schnittstelle kann durch eine automatische Rechtschreibkorrektur ergänzt werden, wie diese in kommerziellen Textverarbeitungen heute üblich sind. Die vermutlich falsch geschriebenen Worte könnten z.B. unterstrichen werden. Natürlich könnte auch sofort ein Klärungsdialog kommen, in dem gefragt wird, ob das Wort richtig geschrieben ist.

Die Schnittstelle kann leicht so abgeändert werden, dass die Datenbank auf einem zentralen Rechner betrieben wird, und der Client ist dann über ein Netzwerk, z.B. das Internet, mit dieser Datenbank verbunden. Das größte Problem bei dieser Änderung ist nicht, die Clienten so zu ändern, dass er auf einen Datenbank auf einem fremden Rechner zugreift[30], sondern die Datenbank und den Clienten so abzusichern, dass durch die Kommunikation keine Sicherheitslücke im Clienten-Computer oder sogar im Netzwerk entsteht.

Im Prinzip ist das Konzept der natürlichen Sprache, verbunden mit Dialogen, optimal dafür geeignet, mit gesprochener Sprache kombiniert zu werden. Aber das Programm bedarf aus meiner Sicht einiger umfassender Änderungen, um mit gesprochener Sprache arbeiten zu

30 Dazu könnte es möglicherweise schon reichen eine IP-Adresse statt localhost als Adresse der Datenbank anzugeben

können. Einige wichtige Unterschiede wurden schon in den Kapiteln 2.2 und 2.4 aufgezeigt. Es ist zwar möglich, das System mit einer Spracherkennung zu kombinieren, für gute Ergebnisse muss das Programm aber angepasst werden.

Die Nutzung der gesprochenen Sprache ist interessant, da das Programm so auch in der natürlichen Umgebung simuliert werden kann. Das Programm kann dann über das Telefon betrieben werden, für Benutzertest wäre somit ein weiter Vergleichstest mit einer richtigen Telefonauskunft möglich.

Ob eine Präsentation der Ergebnisse als ganze Sätze und in einer Oberfläche wie der bisher verwendeten sinnvoll ist, werde ich noch diskutieren, nachdem die Ergebnisse der Evaluation vorgestellt wurden.

4.6.10 Datenschutz

Einige der in diesem Programm und in den hier erwähnten Erweiterungen möglichen und angedachten Suchmöglichkeiten gehen sicherlich weiter, als es der Datenschutz erlaubt. Solange nur fiktive Daten in der Datenbank vorhanden sind, ist dieses Thema sicherlich nicht von Bedeutung. Wenn aber reale Personen gespeichert werden, muss geprüft werden, welche Suchmöglichkeiten überhaupt erlaubt sind.

4.6.11 Automatisches Testen des Programms

Zum besseren Testen wäre es sinnvoll, den Input auch aus einer Datei lesen zu können, damit das Testen bei Änderungen im Programm automatisiert werden kann, wie es Banwart und Berrettoni (1995) vorschlagen. Das System wird nach einer Änderung mit einer vorgefertigten, bei jedem Test gleichen, Menge von Aufgaben gefüttert, und die Ausgabe erfasst. Die hierzu nötige Protokollierung der Systemreaktion existiert ja bereits. Das Testprogramm soll in der Lage sein, die Tests so weit automatisch durchzuführen, dass sie auch über Nacht ablaufen können.

4.7 Zusätzliche Programme

Während der Entwicklung des Programmes ist noch ein weiteres Programm entstanden, um die für das Programm benötigte Datenbank zu erstellen und zu erweitern. Im folgenden wird der Generator für Datenbankeinträge kurz beschrieben.

Zusätzlich zum hier entwickelten Programm ist noch ein konventionelles Programm entstanden, mit dem die Datenbank abgefragt werden kann. Dieses kann möglicherweise in

späteren Studien für eine vergleichende Evaluation der Schnittstellen genutzt werden. Auch von diesem Programm folgt eine kurze Beschreibung.

4.7.1 Der Generator

Der Generator dient zum Erstellen einer Datenbank für das Telefonauskunftsprogramm. Die Daten können zufällig erzeugt oder vom Benutzer eingegeben werden. So wurden z.B. die Daten für die Evaluation mit dem Generator erzeugt. Dazu bietet der Generator folgende Optionen:

1. Person erstellen
2. Firma erstellen
3. Abteilung erstellen (wird von dieser Version des Programmes nicht genutzt)
4. Arbeitsamt – Eine Person einer Abteilung zuordnen (wird auch noch nicht genutzt)
5. Ort erstellen
6. Gewerbe erstellen

4.7.1.1 Person erstellen

In dem Menü „Person erstellen“ gibt es jeweils eine Liste zur Auswahl eines Vornamens, eines Nachnamens, einer Straße und eines Ortes.[31] Der Vorname, der Nachname und die Straße können aber nicht nur aus der Liste ausgewählt, sondern auch in ein Textfeld unter der jeweiligen Liste eingegeben werden. Beim Vornamen ist noch zusätzlich nach dem Namen „(m)“ für Männer und ein „(w)“ für Frauen einzugeben. Doppelnamen müssen ohne Leerzeichen eingegeben werden (z.B. „Müller-Lüdenscheidt“ und nicht „Müller – Lüdenscheidt“[32]).

Zusätzlich zu den schon genannten Feldern gibt es noch 2 Textfelder, in welche die Telefonnummer und die Hausnummer eingetragen werden müssen. Beide Werte werden vom Generator selbständig mit Zufallswerten belegt, können aber natürlich vom Benutzer geändert werden. Der Zufallswert für die Telefonnummer ist eindeutig, d.h. es gibt diese Nummer nicht noch einmal in der Datenbank. Bei den vom Benutzer eingegeben Werten wird dies nicht überprüft.

31 Die Listen mit den Namen sind Textdateien die manuell verändert und erweitert werden können. Die Dateien liegen im selben Verzeichnis wie der Generator. Die Dateinamen sind „vornamen.txt“, „nachnamen.txt“ und „strassen.txt“. Die einzelnen Einträge in den Dateien sind durch Return zu trennen. Vor den Vornamen muss das Geschlecht angegeben werden (m oder f) und mit Komma vom Namen getrennt werden.

32 Beim Auskunftsprogramm ist dies nicht notwendig.

Der Knopf „Zufällig auswählen“ wählt zufällig aus den Listen Vorname, Nachname, Straße und Ort jeweils einen Eintrag aus und belegt die Textfelder Hausnummer und Telefonnummer mit Zufallswerten.

Der Knopf „In Datenbank einfügen“ fügt einen Datensatz mit in die Datenbank ein. Der Datensatz besteht aus den in den Listen markierten Werten, bzw. den in den Textfeldern eingetragen Werten. Sollte in den Textfeldern unter Vorname, Nachname und Straße etwas eingetragen sein, wird dieser Wert statt dem in der Liste ausgewählten Wert in die Datenbank eingefügt.

Nach dem Einfügen werden werden alle Werte wieder neu zufällig ausgewählt. Einträge in den Textfeldern für Vor-, Nachname und Straße werden aber nicht gelöscht, so das sie vom Benutzer weiterverwendet werden können.

Der Knopf „Zurück“ schließt diese Menü und läßt nur noch das Hauptmenü offen.

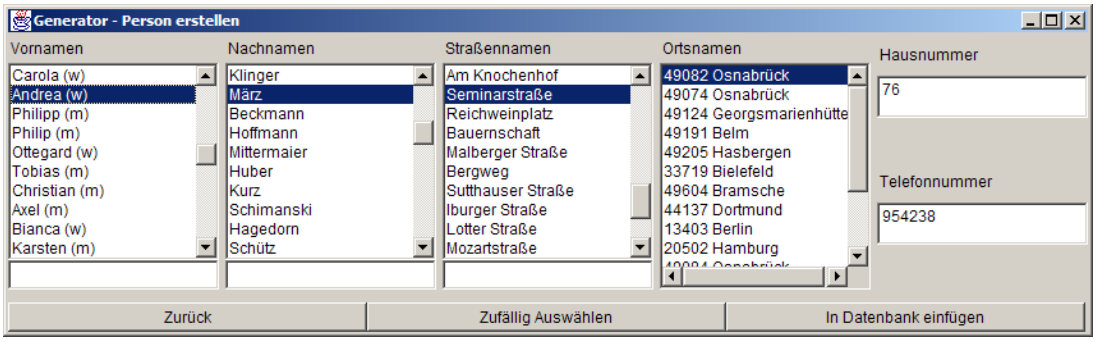


Abbildung 4.6 Das Menü um Personen zu erstellen

4.7.1.2 Firma erstellen

Dieses Menü ist fast identisch mit „Person erstellen“. Statt dem Vor-, bzw. Nachnamefeld gibt es ein Textfeld für den Firmennamen und eine Liste für das Gewerbe in dem die Firma tätig ist. Beim zufälligen Auswählen wird der Firmenname aus dem Namen des Gewerbes und der Endung „GmbH“ erzeugt. Dieser Name kann natürlich vom Benutzer geändert werden.

4.7.1.3 Abteilung erstellen

Diese Option wird vom Auskunftsprogramm nicht genutzt, ist aber im Generator noch vorhanden. Auch in der Datenbank gibt es noch Felder für die Abteilungen. Im Hauptprogramm ist das Suchen nach Abteilungen aus Zeitgründen nicht implementiert worden.

Die Eingabe für die Adresse der Abteilung ist wie bei Person oder Firma erstellen. Zusätzlich muss die Abteilung noch einer bereits existierenden Firma zugeordnet werden, der Abteilung muss eine Funktion zugeordnet werden (z.B. Einkauf oder Vorstand) und die

Abteilung muss benannt werden (Standarteinstellung ist die Bezeichnung der Funktion der Abteilung).

Drückt man auf den Knopf „Aktualisieren“ werden in einer Liste alle Abteilungen angezeigt, welche die ausgewählte Firma besitzt, und es wird zusätzlich die Bezeichnung der ausgewählten Funktion in das Textfeld für den Abteilungsnamen kopiert. Die Adresse der ausgewählten Firma wird als Adresse der Abteilung eingetragen, nur die Telefonnummer wird neu erzeugt.

Wie in den anderen Menüs auch gibt es wieder einen Knopf, um den hier zusammengestellten Datensatz in die Datenbank einzufügen.

4.7.1.4 „Arbeitsamt“

Aus den selben Gründen, weshalb die Abteilungen nicht in der Auskunft implementiert sind, wurden auch die Berufe nicht aufgenommen.

Hier gibt es in einer Liste alle Personen, denen noch keine Abteilung und kein Beruf zugeordnet wurde. Dann eine Liste mit den Abteilungen aller Firmen und eine Liste mit Berufen. Zusätzlich zwei Textfelder für die Telefonnummer und den Raum. Der Raum braucht keine Zahl zu sein, sondern kann auch aus Buchstaben bestehen. Wenn der Raum zufällig belegt wird, ist es aber eine Zahl.

Das zufällige Belegen und Einfügen in die Datenbank verhält sich wie in den anderen Menüs auch.

4.7.1.5 Ort erstellen

In diesem Menü gibt es eine Liste, in der alle schon in der Datenbank eingetragen Orte angezeigt werden. Was in dieser Liste ausgewählt wird, hat keine Bedeutung. Die Liste wird nur angezeigt, damit der Benutzer nachgucken kann, welche Orte er schon eingetragen hat.

Eingetragen in die Datenbank werden 3 Textfelder. Eines für den Ortsnamen, eines für die Postleitzahl und eines für die Vorwahl. In den Feldern für Postleitzahl und Vorwahl werden nur Zahlen akzeptiert. Alle Felder müssen belegt sein, damit ein Datensatz in die Datenbank eingefügt werden kann.

4.7.1.6 Gewerbe erstellen

In diesem Menü gibt es eine Liste mit allen vorhanden Gewerbearten. Um ein neues Gewerbe zu erstellen, wählt man diesen Menüpunkt aus dieser Liste. Wählt man ein schon

vorhandenes Gewerbe aus dieser Liste, dann ist das Gewerbe, das man nun in die Datenbank einträgt bedeutungsgleich mit dem Gewählten (z.B. Bank und Sparkasse).

Das neue Gewerbe muss inklusive einiger konjugierter Formen eingegeben werden. Zuerst einmal kann aber eine allgemeine Bezeichnung für dieses Gewerbe eingegeben werden. Wenn hier nichts eingegeben wird, dann wird die nominativ singular Form als Bezeichnung benutzt. Es muss der Genus des Wortes aus einem Menü gewählt werden.

Es muss der Nominativ Singular, und der Nominativ Plural eingegeben werden. Der Genitiv Singular, Dativ Singular und Akkusativ Singular müssen nur eingegeben werden, falls sie von Nominativ Singular abweichen. Der Dativ Plural muss auch nur eingegeben werden, falls er von Nominativ Plural abweicht.

Es brauchen nie alle möglichen Formen eingegeben werden, da der Genitiv Plural und Akkusativ Plural immer aus dem Nominativ schließen kann. Der Akkusativ Singular wird sehr selten genutzt (Duden Bd.4, S. 222ff).

Wie in allen anderen Menüs muss ein neuer Datensatz in die Datenbank eingefügt werden. Anders als bei allen anderen Daten muss hier noch eine Prologdatei erstellt werden, damit das Auskunftsprogramm auf die Gewerbedaten zugreifen kann. Dazu muss man einmal, nachdem man neue Gewerbe in die Datenbank eingetragen hat den Knopf „Prologdatei erstellen!“ drücken.

4.7.2 Die konventionelle Schnittstelle

Um einen einfach zu programmierenden und in der Entwicklungsphase verlässlichen Zugriff auf die Datenbank zu haben, ist ein Programm entstanden mit dem man auf herkömmliche Weise in der Datenbank suchen kann. Dieses Programm kann auch später für eine vergleichende Evaluation der natürlichsprachlichen Schnittstelle benutzt werden. In dieser Arbeit wird ein solcher Vergleich aber nicht beabsichtigt.

Die Entwicklungszeit einer solchen Schnittstelle ist drastisch kürzer als die einer natürlichsprachlichen Schnittstelle (Stunden zu Wochen).

Das Programm orientiert sich an der Internetprogramm der Deutsche Telekom unter www.telefonbuch.de, Version vom Juli 2001 war. Erweitert ist es um die Option, nach Firmen und ähnlichen Namen[33] zu suchen.

Das Programm hat 2 Masken, zwischen denen mit dem Knopf „erweiterte Suche“ / „einfache Suche“ umgeschaltet werden kann. In der einfachen Suche gibt es ein Textfeld für

33 Bei der aktuellen Version von telefonbuch.de (Januar 2002) ist jetzt die Suche nach ähnlichen Namen in der Detail-Suche möglich.

den Nachnamen bzw. Firmennamen und ein weiteres Textfeld für den Ort oder die Postleitzahl. In dieser Maske ist es notwendig, einen Namen einzugeben. Die Suche kann gestartet werden, indem man den Knopf „Suche“ drückt oder in dem Namensfeld Return drückt. In einer Checkbox kann man wählen, ob auch ähnliche Namen gesucht werden sollen.

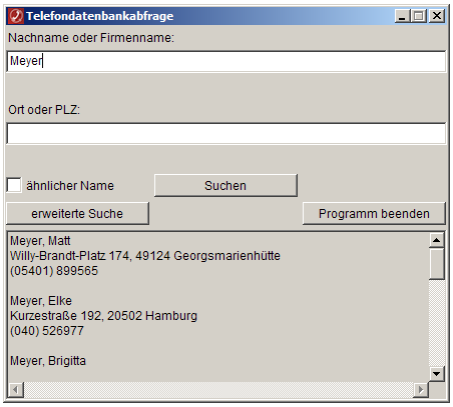


Abbildung 4.7Die Maske für die einfache Suche

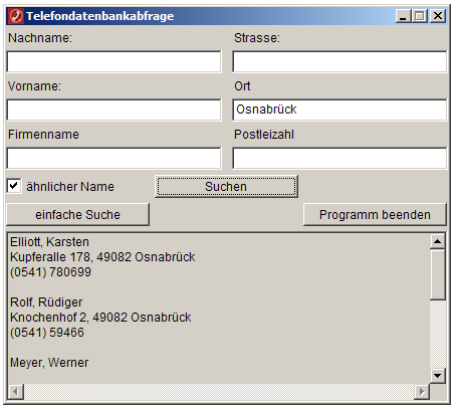


Abbildung 4.8Die Maske für die erweiterte Suche

Die erweiterte Suche hat getrennte Felder für jeden möglichen Eintrag. Angegeben werden kann der Vorname, der Nachname, der Firmenname, die Straße (gegebenenfalls mit Hausnummer), der Ort und die Postleitzahl. In den Namensfeldern kann die Suche auch mit Return gestartet werden, ansonsten muss der Suche-Knopf gedrückt werden. Wenn man diese Maske verwendet, reicht es, wenn ein beliebiges Feld ausgefüllt wurde, um eine Suche auszuführen. Auch in diesem Modus ist natürlich die Checkbox für die Suche nach ähnlichen Namen vorhanden.

Die Ausgabe erfolgt in eine TextArea unter dem Suchfeld. Die Eingaben des Benutzers werden auch protokolliert, wenn dieser eine Suche startet. Das Protokoll ist in der Datei „abfrage-protokol.txt“ im Arbeitsverzeichnis gespeichert. Es wird im Protokoll zusätzlich nur festgehalten wann die Suche gestartet wurde.

4.8 Technisches

Das Programm ist in Java 2 mit dem JDK 1.3 geschrieben worden. Teile der Parsers sind in Prolog. Als Prolog Interpreter wird SWI-Prolog 4 verwendet. Die Anbindung der Prologcodes an Java erfolgt über JPL 1.0. JPL nutzt das Java Native Interface (JNI).

Als Datenbank wird MySQL 3.23.39 eingesetzt. Die Anbindung der Datenbank an Java erfolgt über JDBC als Treiber wird MM.MySQL 2.0.4 eingesetzt.

Das Programm wurde für Windows 9X und NT compiliert. Für einen Einsatz unter Linux müsste wahrscheinlich nur JPL noch einmal compiliert werden.

Ein Fehler im JPL scheint einige wenige Klauseln in Prologcode zu ignorieren, was sich auch an der Größe der compilierten Dateien feststellen läßt. In diesem Programm sind die Folgen aber kaum zu bemerken[34]. Der auffälligste Fall ist, dass die Klausel „Nachname, Vorname“ ignoriert wird, was durch die Analyse in Java weitestgehend abgefangen wird.

34 Auch wenn zahlreiche bisher ungeklärte Probleme möglicherweise in Beziehung zum JPL stehen. Den Fehler aber generelle auf diese Schnittstelle zu schieben wäre kontraproduktiv.

5 Ergebnis der Evaluation

Diese Evaluation wurde von Jens Schroeder und Sarah Helmich im Rahmen eines Forschungspraktikums durchgeführt. Sie benutzten die Methode der Videokonfrontation in dieser Studie.

Schroeder und Helmich stellen leider nur fast das von Banwart und Berrettoni (1995) geforderte unabhängige Team zum Testen eines Softwareproduktes dar, da Jens Schroeder schon recht früh in Designentscheidungen des Telefonauskunftsprogrammes mit einbezogen wurde.

Es wurde die Methode der Videokonfrontation eingesetzt. Dabei werden die Versuchspersonen gebeten, eine Reihe von Szenarien durchzuführen, während sie dabei auf Video aufgezeichnet werden. Aufgezeichnet werden das Gesicht des Probanden, die Tastatur und das Monitorbild.

Nachdem die Aufgaben bearbeitet wurden schaut sich die Versuchsperson zusammen mit dem Versuchsleiter das Video an. Dabei wird die Versuchsperson gebeten sein Vorgehen zu kommentieren. Der Versuchsleiter fragt gegebenenfalls, wenn nach seiner Ansicht das im Video gezeigte Verhalten der Versuchsperson nicht eindeutig ist und kommentiert werden sollte. Zwischen der Aufgabenbearbeitung und dem Interview wurde den Versuchspersonen noch ein Fragebogen vorgelegt in dem sie das Programm nach hedonistischen und ergonomischen Aspekten bewerten sollten.

Die Ziele der Videokonfrontation sind, die Aufdeckung von Gestaltungsmängeln der Software, die Identifizierung von Mängeln bei der Anpassung von Softwaresystemen an die Aufgabenabläufe und Bearbeitungspraxis der Nutzer und praktische Hinweise für eine handlungsbezogene Gestaltung von Oberflächenelementen zu erhalten.

In der Analyse der mit dieser Methode gewonnenen Daten werden subjektive Strukturen der Aufgabenbearbeitung, handlungsstrukturierende Kognitionen und handlungsbegleitende Emotionen berücksichtigt.

In diesem Test mussten die Versuchspersonen 3 Szenarien bearbeiten. Im ersten sollte ein spezieller Elektromarkt und danach nach Elektromärkten im Allgemeinen gesucht werden. Im zweiten Szenario wurde eine spezielle Person gesucht, bei der aber nicht klar ist, in welchem von 2 Orten sie wohnt. Und im dritten Szenario sollte zuerst allgemein nach einem

griechischen Restaurant gesucht werden und danach nach einem speziellen italienischen Restaurant.

Im anschließenden Interview fragt der Versuchsleiter zuerst, welchen Gesamteindruck das Programm auf den Probanden gemacht hat. Für die daran anschließende Betrachtung des Videos werden dem Probanden folgende Leitfragen genannt:

- Was fanden sie besonders hilfreich?
- Was hat ihnen Probleme bereitet?
- Was müsste man ihrer Meinung nach verbessern?

Und in Beziehung auf die Dialoge in dem Programm:

- War der Dialog verständlich?
- Wenn nicht, wie hätten sie sich den Dialog vorgestellt?

Diese Untersuchung wurde mit neun Teilnehmern durchgeführt, vier weiblichen und fünf männlichen Studenten. Acht der Teilnehmer hatten schon mehr als zwei Jahre PC-Erfahrungen und ein Teilnehmer über ein halbes Jahr.

Alle Teilnehmer haben schon Erfahrungen mit Suchmaschinen im Internet und in Bibliotheken.

5.1 Deskriptive Merkmale

Es wurden von den Benutzern 1366 Worte eingegeben. Diese Eingabe besteht aus 161 unterschiedlichen Worten, 54 davon sind unterschiedliche Eigennamen, Firmennamen, Straßennamen und ähnlichem. Die restlichen 92 Worte müssen lexikalisch erfasst sein. Diese 92 Worte lassen sich auf 70 Stammformen reduzieren, von denen 28 nicht vom Lexikon des Telefonbuchprogrammes erfasst sind.

Bei den Worten sind insgesamt nur 12 Rechtschreibfehler gemacht worden was einer Fehlerquote von 0,89% entspricht.

Die Eingaben der Versuchspersonen sind durchschnittlich 9,17 Worte lang. Die kürzeste Eingabe ist ein Wort lang, die maximale Länge ist 16 Worte.

Die vom System erzeugten Texte sind durchschnittlich 10,5 Worte lang, wobei die maximale Länge von 359 Worten sich durch die Ausgabe einer Liste von Anschriften erklären läßt.

Insgesamt gab es 572 Dialogzüge, 311 vom Benutzer und 261 vom Programm. Es sind aber nur 376 Dialogzüge (220 Benutzerzüge und 156 Systemzüge) interessant, da es zu vielen Eingabewiederholungen kam, wenn z.B. „neue Anfrage“ mehrmals hintereinander gedrückt

wurde oder der Benutzer seine Eingabe wiederholte, wenn das System noch rechnete bzw. bei der Verarbeitung stecken geblieben war.

Auf alle Szenarien bezogen		
	absolut	%
erreicht	18	40,00
teilweise erreicht	6	13,33
nicht erreicht	21	46,67
Summe	45	100,00

Tabelle 5.1 Gesamtwerte der Aufgabenbearbeitung

Von insgesamt 45 Aufgaben und Teilaufgaben, die die Versuchspersonen zu bearbeiten hatten wurden 18 komplett gelöst, was einer Quote von 40 % entspricht. 6 mal (13,33%) wurde die Aufgabe teilweise gelöst und bei 21 Aufgaben (46,67%) der Aufgaben gelang es den Versuchspersonen überhaupt, nicht die Aufgabe zu lösen.

Prozentuale Anteile				
	erreicht	teilweise erreicht	nicht erreicht	Summe
Szenario 1				
Teilaufg. 1	44,44	11,11	44,44	100,00
Teilaufg. 2	0,00	44,44	55,56	100,00
Szenario 2				
Aufgabe 1	22,22	0,00	77,78	100,00
Szenario 3				
Teilaufg. 1	66,67	0,00	33,33	100,00
Teilaufg. 2	66,67	11,11	22,22	100,00

Tabelle 5.2Aufgabenbearbeitung nach Szenarien

Über die einzelnen Szenarien verteilt sieht es so aus, dass das dritte Szenario „Restaurantsuche“ den Versuchspersonen am wenigsten Probleme bereitete, 66,67% haben beiden Teilaufgaben erfolgreich bearbeitet, bei der zweiten Teilaufgabe gelang es auch noch einer Versuchsperson (11,11%) die Aufgabe zumindest teilweise zu lösen. Das Zweite Szenario, die Suche nach einer Person, deren Wohnort nicht genau bekannt ist, war die Aufgabe, die die größten Schwierigkeiten bereitete, nur 22,22% der Probanden konnte diese Aufgabe lösen, eine teilweise Lösung gab es hier überhaupt nicht. Aber auch der zweite Teil des ersten Szenarios „Elektromarkt“ konnte von Niemandem komplett gelöst werden. 44,44%

konnten dieses Ziel aber wenigstens teilweise erreichen. Der erste Teil dieses Szenarios wurde von 4 Personen (44,44%) komplett und von einer Versuchsperson (11,11%) teilweise gelöst.

DN Anz. Anfragen	
Szenario 1	**Mittelwert**
Teilaufg. 1	3,11
Teilaufg. 2	9,75
Szenario 2	
Aufgabe 1	8,89
Szenario 3	
Teilaufg. 1	4,67
Teilaufg. 2	4,22

Tabelle 5.3 Durschnitlliche Anzahl der Benutzeranfragen pro Teilaufgabe

Es ist festzustellen das die erste Teilaufgabe des ersten Szenarios mit den wenigsten Anfragen gelöst werden konnte (3,11), wo hingegen auf die zweiten Teilaufgabe mit 9,75 die meisten Anfragen entfallen.

Auch für das zweite Szenario wurden mit 8,89 sehr viele Anfragen benötigt, wo hingegegen die beiden Teilaufgaben des dritten Szenarios mit 4,67 bzw. 4,22 wieder deutlich schneller gelöst werden konnten.

Es ist klar zu sehen das die Anzahl der Anfragen in Relation zur erfolgreichen Bearbeitung der Aufgabe steht. Die Aufgaben die fast nicht oder gar nicht von den Probanden gelöst werden konnten erzeugten die meisten Anfragen, da die Versuchspersonen mehrere Versuche unternahmen bevor sie die Bearbeitung abbrachen.

Anfragen gesamt	210	100%
davon "ich" Form	102	48,57%
davon "Sie" Form	10	4,76%
davon Schlagwortanfragen	43	20,48%
Anfragen sonstige	55	26,19%

Tabelle 5.4 Art der Anfragen

Von den 220 Benutzerzügen sind nur 210 für eine weitere Auswertung verwertbar. Von diesen 210 Zügen sind 48,57% in der ersten Person gehalten, davon entfallen die meisten Sätze auf die Phrase „Ich suche...“ die mit 40,48% aller Eingaben des Benutzers verwendet wird.

20,48% der Eingaben bestehen aus Schlagworten und 4,76% sind in der Sie-Form verfasst, z.B. „Haben Sie ...“. Weitere 26,19% der Eingaben entfallen auf sonstige Formulierungen wie z.B. „Wie lautet...“.

Form der Ansprache	Häufigkeit	% von NL-Anfragen gesamt
1 = Ich suche	85	40,48%
2 = In welcher	1	0,48%
3 = Ich brauche	4	1,90%
4 = Ich möchte	1	0,48%
5 = Haben Sie noch	3	1,43%
6 = Gibt es	4	1,90%
7 = Wo finde ich	3	1,43%
8 = Ich hätte gerne	4	1,90%
9 = Wie lautet	16	7,62%
10 = Ich möchte	2	0,95%
11 = Können Sie mir	1	0,48%
12 = Suche	6	2,86%
13 = Ich benötige	1	0,48%
14 = Bitte geben Sie mir	2	0,95%
15 = Unter welcher	1	0,48%
16 = Geben Sie mir	4	1,90%
17 = Wie heißt das	1	0,48%
18 = Welche	1	0,48%
19 = Ich möchte gerne	2	0,95%

Tabelle 5.5 Häufigkeiten der Form der Benutzeranfragen

Nur ein Proband versuchte, vom System eine vorzeitige Ergebnissausgabe zu erhalten und damit den Dialog abzukürzen. Dies scheiterte aber an der vom System nicht erkannten Wortwahl.

5.2 Ergebnisse aus der Videokonfrontation

In diesem Abschnitt werden die Ergebnisse vorgestellt, die Schroeder und Helmich durch die Videokonfrontation erhalten haben.

5.2.1 Datenausgabe

Die meisten Anmerkungen machten die Versuchspersonen zum Thema Datenausgabe. Zur Ausgabelogik wurde das meiste gesagt, wobei sich positive und negative Kommentare die Waage halten. Die meisten negativen Aussagen wurden zum Thema Feedback getätigt.

5.2.1.1 Feedback

Das Fehlen von Feedback wurde bemängelt. Alle Probanden gaben an, dass eine große Unzufriedenheit daraus resultierte, dass das System oft nach einer Anfrage keine Reaktionen mehr zeigte. Es konnte nicht erschlossen werden, ob das System arbeitet oder sich aufgehängt hat

Die Rückmeldungen und Fehlermeldungen wurden von den Probanden als zu lang empfunden. Ein Teilnehmer sagte sogar, dass er die Meldungen wegen der Länge nicht weiter durchgelesen hatte.

Im Weiteren wurden bemängelt, dass die Rückmeldungen keine Optionen anbieten, was als nächstes zu tun ist. So hatten einigen Probanden das Problem, dass sie keine Ideen hatten, wie sie die nächste Eingabe formulieren sollten.

Das System sollte generell Feedback geben. Ein Vorschlag war eine Textrückmeldung, was das System macht, wie z.B. „es dauert einen Moment“ oder „bitte warten“.

Eine andere Anregung betraf die Visualisierung des Feedbacks. Die Probanden wünschten sich z.B. dass eine Sanduhr oder Statusbalken den Arbeitsverlauf des Systems anzeigt.

Bei fehlerhaften Eingaben hatten die Probanden Probleme mit den Rückmeldungen des Programmes, aus denen sie oft nicht erkennen konnten, was nicht korrekt war. Das System sollte explizit melden, was nicht korrekt verarbeitet werden konnte oder was fehlt. Es fehlte hier die Transparenz für den Benutzer.

5.2.1.2 Dialogqualität

Die Versuchspersonen empfanden es als störend, das oft nach dem gesuchten Namen gefragt wurde. Zum Beispiel: Wenn man nach einem Gewerbe sucht, wurde oft nach einem Firmenname gefragt, der den Probanden nicht bekannt war, da dieser gesucht wurde.

Ein Proband empfand es als „Schikane“, dass das System zwar die Information gibt, Ergebnisse gefunden zu haben, aber das System keine Handlungsoption gibt, um diese Ergebnisse anzuzeigen.

Die Nachfragehierarchie verwirrte einen Probanden. Er wünschten sich eine nachvollziehbare Nachfragelogik, z.B. erst Nachfrage nach Ort, Straße, etc.

Eine andere Anregung war, dass das System bei gefundenen Ergebnissen gleich das Ergebnis ausgibt, und nicht noch weiter Nachfragen stellt, um die Ergebnismenge zu verringern.

5.2.1.3 Sprachverständnis

Es wurde von einem Probanden bemängelt, dass vom Nutzer korrekte Rechtschreibung gefordert wird, aber vom System fehlerhafte Ausgaben kommen („ korregieren“).

5.2.1.4 Ausgabelogik

Allgemein würde es begrüßt, wenn nach einer Eingabe sofort ein Ergebnis bzw. eine Meldung erscheint. Bei einigen Anfragen waren die Probanden angenehm überrascht, dass auch bei allgemein gehaltenen Anfragen ein Ergebnis ausgegeben wurde.

Ein Proband war der Meinung, dass die Informationsfülle der Ausgabe passend ist.

Dass bereits präsentierte Ergebnisse sich nicht reproduzieren ließen, missfiel einigen Probanden. So kommentierte eine Person, man hätte das Gefühl, das Programm „lüge“, da er ja wusste, dass die Adresse in der Datenbank vorhanden war.

Von allen Probanden wurde es als unangenehm empfunden, dass die Anfragen vom System oft nicht verstanden wurden. Besonders dann, wenn das System eine Fehlermeldung ausgab, in der Textelemente der Anfrage als ein fehlerhafter Name verstanden wurde (z.B. Ortsname als Firmenname).

Die Systemrückmeldung: „Leider wurde kein Eintrag gefunden. Eine Person mit dem Nachnamen Klinger gibt es nicht in 49082 Osnabrück, 49074 Osnabrück oder 49084 Osnabrück .“ sollte nicht so formuliert werden, in welchen PLZ-Bereich ein „Klinger“ nicht zu finden ist. Es sollte nur dann eine PLZ ausgeben werden , wenn dort ein „Klinger“ zu finden ist.

Eine andere Ausgabe, die auf eine konkrete Anfrage 28 Einträge anzeigte, wurde von einigen Probanden als verwirrend und unübersichtlich empfunden.

Bei allgemeinen Anfragen (z.B. „suche Restaurant“) sollten alle Ergebnisse sofort ausgegeben werden.

Im weiteren wurde angeregt, bei der Ausgabe der Ergebnisse die komplette Anschrift mit Telefonnummer anzuzeigen, da es sonst Probleme gibt, die ausgegebenen Nummern der Restadresse zuzuordnen.

5.2.2 Designaspekte

Es wurde als störend empfunden, dass der Cursor bei einer neuen Anfrage nicht im Texteingabefeld steht.

Ein Proband bemerkte, dass mal die letzte Eingabe im Eingabefeld stehen bleibt, und mal nicht. Dies sei verwirrend.

Eine andere Versuchsperson fand es hilfreich, dass die letzte Eingabe im Eingabefeld stehen bleibt. Das erleichtert das Abändern der Eingabe.

Ein Proband regte an, einen Button „Suche starten“ einzubauen, da die Eingabebestätigung mit der <Enter> - Taste nicht sofort klar war.

Eine andere Anregung war, dass man ein Button „ Ergebnisse anzeigen“ mit einbauen sollte, damit der Nutzer schnell die Möglichkeit hat, die Suche zu verkürzen.

5.2.3 Eingabeverarbeitung

5.2.3.1 Eingabeart

Von einige Probanden wurde die Schlagwortsuche eingesetzt, da dies eine bekannte und erfolgsversprechende Strategie ist, an die die Probanden schon aus dem Internet gewöhnt sind. Die Schlagwortsuche wurde vermehrt eingesetzt, nachdem die Versuchspersonen mit natürlichsprachlichen Versuchen gescheitert waren.

Einige Probanden versuchten, die Möglichkeiten der NL-Schnittstelle bewusst auszutesten, z.B. durch lange Eingaben oder durch Eingabe von zwei Sätzen. Dies geschah in der Erwartung, das diese Äußerungen nicht verarbeitet werden können.

Eine Person probierte zusätzliche eine boolsche Eingabeform, um an ein Ergebnis zu gelangen, aber ohne Erfolg.

5.2.3.2 Handlungsstrategie

Alle Probanden hatten Schwierigkeiten, im Dialogverlauf einer Anfrage passende Formulierungen zu finden, die das System verarbeiten kann. Die Probanden haben nach kurzer Interaktion mit der Software Strategien entwickelt, um möglichst passende Anfragen zu finden.

Das eingeschränkte Feedback des Systems wurde negativ herausgestellt, da die Probanden nicht genau wussten, was das System benötigt, um die Anfrage zu verarbeiten.

So kam es öfter vor, dass Probanden leicht am System verzweifelten, da sie keine Ideen zu einer Umformulierung finden konnten.

Einige Probanden versuchten mit wiederholten Anfragen ans Ziel zu kommen. Oft war dies Ausdruck von Unzufriedenheit, da das System keine Reaktion zeigte. So wurde schnell nochmals die Eingabetaste ausgelöst, um die gestellte Anfrage noch mal zu starten.

Andere Personen schauten sich die vorherigen Dialogschritte mit dem System an, und versuchten aus dem Dialog zu erschließen, welche Formulierungen vom System angenommen wurden und welche zu meiden waren. Eine weitere Gruppe orientierte sich an der Szenarienformulierung.

5.2.3.3 Eingabeform

Eine Versuchsperson merkte an, dass es selbstverständlich sei, mit einem natürlich-sprachlichen System in persönlicher Form zu kommunizieren.

5.3 Bewertung des Systems durch die Benutzer

Im diesem Abschnitt werden die Ergebnisse des HedoDiff-Fragebogens, zur Erfassung der hedonistischen und ergonomischen Qualitäten der Software vorgestellt.

Die Versuchspersonen sollten auf einer Skala von 1, bester Wert, bis 7, schlechtester Wert, die jeweiligen Items beurteilen.

Der Error Bar in den folgenden Grafiken zeigt zwischen dem oberen und unteren Querbalken eines Items die Daten der Versuchspersonen. Das Vertrauensintervall liegt hier bei 95%.

Schroeder und Helmich kommen zu den folgenden Ergebnissen:

Die Hedonistische Qualität erfasst Systemqualitäten, die nicht der direkten Zielerreichung dienen, z.B. originell, vertraut, innovativ, interessant.

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
interessant/langweilig	9	1	6	3,56	1,878
wertvoll/minderwertig	9	1	6	3,78	1,481
aufregend/ermüdend	9	2	6	4,11	1,537
außergewöhnlich/üblich	9	2	5	3,11	1,269
eindrucksvoll/unauffällig	9	1	7	4,00	1,871
originell/gewöhnlich	9	1	5	2,67	1,118
innovativ/konservativ	9	1	5	2,44	1,236
Valid N (listwise)	9				

Tabelle 5.6 Hedonistische Qualität des Systems

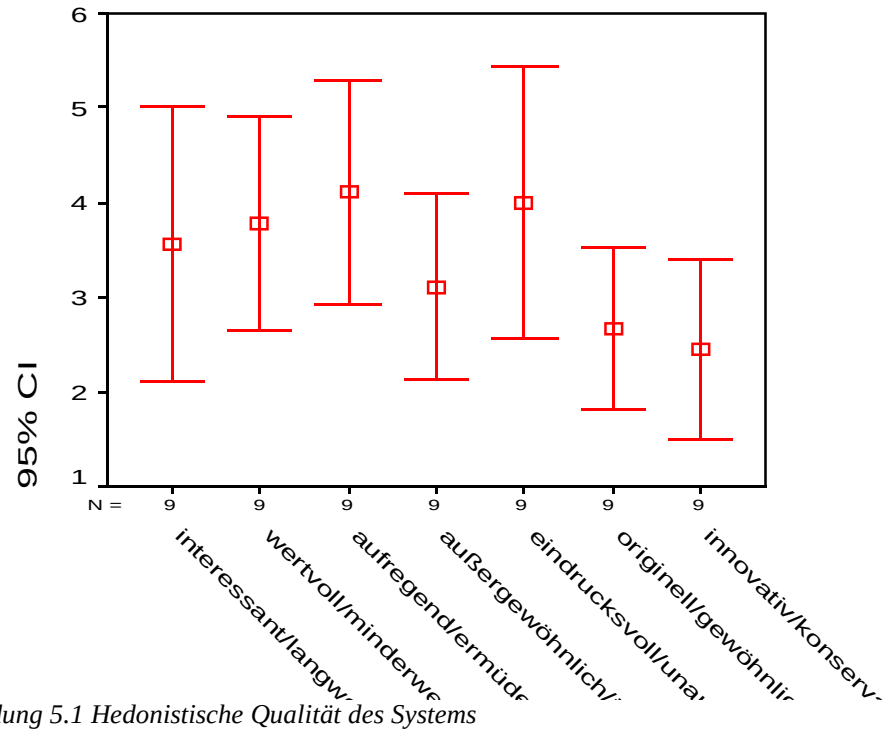


Abbildung 5.1 Hedonistische Qualität des Systems

Die Versuchspersonen bewerteten die hedonistische Qualität des Systems nicht besonders gut (M=3,38 SD=1,56). Die Stärken des Programms liegen in der Originalität und Innovation. Das natürlichsprachliche Dialogsystem wird trotzdem als üblich und unauffällig

wahrgenommen, was unter anderem an der schlichten grafischen Oberflächengestaltung liegen kann.

Die ergonomische Qualität erfasst die traditionelle Usability, d.h. es werden Systemqualitäten erfasst, die der direkten Zielerreichung, der Aufgabenerfüllung dienen.

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
verständlich/unverständl.	9	2	7	3,67	1,803
unterstützend/behind.	9	2	6	3,89	1,833
einfach/kompliziert	9	2	5	3,56	1,130
voraussag./unberechen.	9	2	7	4,89	1,616
übersichtlich/verwirrend	9	2	7	4,11	2,088
vertrauensw./unseriös	9	1	5	3,11	1,453
kontrollier./unkontrollier.	9	2	6	4,78	1,302
vertraut/fremd	9	2	7	4,22	1,856
Valid N (listwise)	9				

Tabelle 5.7 Ergonomische Qualität des Systems

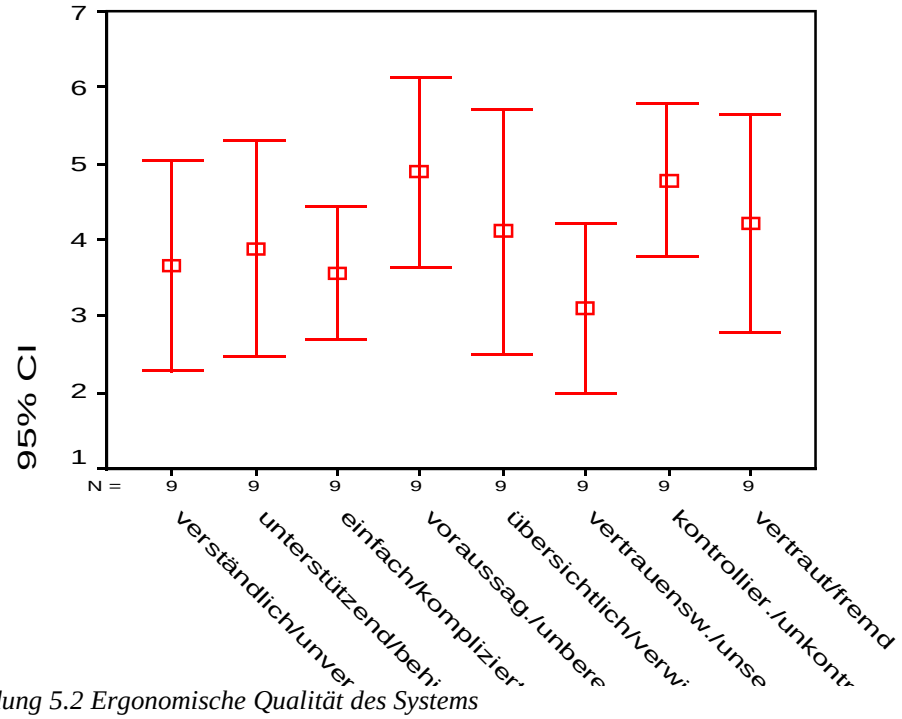


Abbildung 5.2 Ergonomische Qualität des Systems

Die ergonomische Qualität wird relativ schlecht beurteilt (M=4,3 SD=1,678). Aufgrund der vielen Abstürze, Aufhänger und teilweise unverständlicher Nachfragen wurde das System wohl

als unberechenbar und unkontrollierbar eingestuft. Die Werte von „behindernd“ und „verwirrend“ werden dadurch auch beeinflusst.

Die schon erwähnten hohen Werte für Innovation, lassen möglicherweise erkennen, dass ein System, das mit natürlicher Sprache und Dialogen arbeitet, von den Benutzern möglicherweise noch als befremdlich eingestuft wird.

Die nun folgenden Werte stellen den Appeal des Programms dar. Der Appeal sagt aus, in welchem Ausmaß der Benutzer das System akzeptiert und wie sehr er sich durch das System anregen läßt. Hiermit ist z.B. gemeint, wie ästhetisch, angenehm, gut und einladend der Benutzer das System findet.

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
angenehm/unangenehm	9	2	6	3,44	1,236
gut/schlecht	9	2	6	4,00	1,581
ästhetisch/unästhetisch	9	2	6	3,89	1,054
einladend/zurückweisend	9	2	6	4,00	1,658
anziehend/abstoßend	9	2	6	4,00	1,581
sympathisch/unsympath.	9	2	6	4,22	1,563
motivierend/entmutigend	9	3	7	5,22	1,394
erstrebenswert/reizlos	9	1	5	2,78	1,481
Valid N (listwise)	9				

Tabelle 5.8 Appeal des Systems

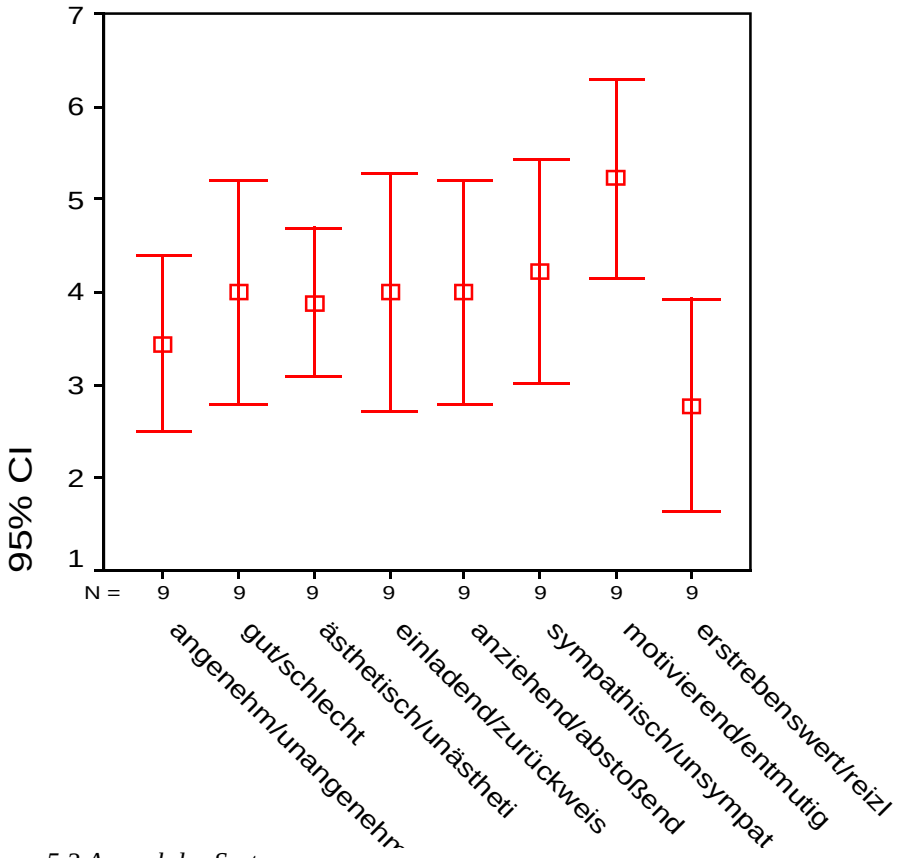


Abbildung 5.3 Appeal des Systems

Die hedonistischen und ergonomischen Qualitäten haben Wechselwirkungen, so dass sie mit vergleichbaren Werten den Appeal beeinflussen (M=3,94 SD=1,528).

Es fällt die schlechte Beurteilung im Bereich der Motivation auf, die sich wahrscheinlich durch die von den Benutzern empfundene Unkontrollierbarkeit des Systems ableiten läßt.

Die Probanden halten das Programm aber trotz einiger schlechter Beurteilungen für erstrebenswert.

5.4 Schlussfolgerungen

Schroeder und Helmich ziehen in ihrer Auswertung der Untersuchung die folgende Schlüsse:

„Wesentliche Aspekte der Datenausgabe betreffen die Dialogfähigkeit, in der Fähigkeit, die Nutzerintentionen zu verstehen und das Feedback.

Die Dialogfähigkeit des Systems war aus drei wesentlichen Gründen eingeschränkt:

1. Zunächst war das Softwaresystem nicht in der Lage, alle erwarteten Funktionsmöglichkeiten umzusetzen. Es konnten keine „OR“ Beziehungen hergestellt werden, was zwangsläufig unter Berücksichtigung der Szenarien zu Fehlern führte.
2. Ein anderes Systemproblem lag in der korrekten Zuordnung der Begrifflichkeiten. Es gelang dem System nicht optimal, die Anfragen semantisch aufzulösen, so dass Systemrückfragen mit falschen Zuordnungen (z.B. Ort als Name) den Benutzer irritierten.
3. Fehlendes Feedback war das zentrale Thema in der Untersuchung. Aufgrund der technischen Gegebenheiten kam es zu Wartezeiten, die vom System nicht kommentiert wurde. Alle Versuchspersonen hatten somit eine starke Unsicherheit, ob das System überhaupt noch reagiert. Diese Eigenschaft der Software war oft negativer Emotionsauslöser, die sich später auch negative im semantischen Differential niederschlug. Eine einfache Visualisierung mittels Statusbalken oder Sanduhr hätte den meisten Probanden genügt. Differenzierter wäre ein schriftliches Feedback wie „bitte warten“.

Bei der Eingabeverarbeitung ist anzumerken, dass die Probanden bei nicht erfolgreichen natürlichsprachlichen Anfragen, die Suchstrategien wechselten. Nach kurzen Wechsel der Eingabeformulierungen wurde schnell auf die klassische Schlagwort-Suchanfragen gewechselt. Die Schlagwort Suchanfragen waren allen Probanden aus Vorerfahrungen im Internet und Büchereisystemen bekannt und vertraut.

Es ist anzumerken, dass es wichtig ist, dass ein natürlichsprachliches System möglichst wenig Fehler produziert und schnell die geforderten Ergebnisse präsentiert, da die Fehlertoleranz der Probanden niedrig ist. Verstärkt wird dies durch fehlendes Feedback vom System.“

6 Diskussion

Es ist natürlich festzustellen, dass es sich bei dem Programm nur um einen Prototypen handelt, ein fehlerfreies ausgereiftes Produkt war in der begrenzten Zeit nicht zu erwarten.

Die teilweise langen Wartezeiten bei der Verarbeitung der Eingabe dürften unter anderem am Prologteil des Parsers und dem Interface zu Prolog liegen. Gelegentlich kann die MySQL Datenbank einige Zugriffsprobleme haben und lange Pausen erzeugen.

Die Auskunfts-Domäne hat sich nicht als optimal erwiesen, da es doch relativ schwierig ist, einen Parser zu entwickeln, der nicht lexikalisch erfasste Einträge zuverlässig erfasst und ihrer jeweiligen Bedeutung zuordnet. Wenn dies mal nicht richtig funktioniert, zeigen die Benutzer Unverständnis darüber, wie ein solcher Fehler überhaupt passieren kann, z.B. dass ein Ortsname als ein Firmenname interpretiert wird.

Die Evaluation zeigt, dass einige der Probleme weder am System noch an einem falschen Verhalten der Benutzer[35] liegen, sondern daran, dass das System noch an das Benutzerverhalten angepasst werden muss.

Viele Benutzer mit Vorerfahrungen benutzen lieber Strategien wie die Schlagwortsuche, oder verfallen zumindest schnell auf diese Strategien, wenn ihre Bemühungen, natürliche Sprache zu benutzen, scheitern. Andere Benutzer sind nicht bereit, sich an das System anzupassen, sondern erwarten, dass das System sich an sie anpasst, auch wenn dies bedeutet, dass sie keinen Erfolg bei ihrer Tätigkeit haben werden. So konnte eine der Versuchspersonen in der Evaluation auch nur eine Teilaufgabe lösen, da sie konsequent bei ihrer Strategie blieb und vom Programm erwartete, dass es sich anpasst.

In der Untersuchung wurden weniger Aufgaben als von mir erwartet korrekt gelöst. Der Hauptgrund dafür dürfte aber nicht in der Funktionalität des Programms an sich liegen, sondern in zahlreichen kleinen Bugs, fehlenden syntaktischen Varianten und fehlenden lexikalischen Einträgen. So war die Funktionalität Begriffe mit „oder“ zu verknüpfen während der Evaluation nicht vorhanden. Wahrscheinlich wurden die hierzu benötigten Methoden, bei der Behebung einiger kleiner Fehler kurz vor der Evaluation, deaktiviert.

35 Falls es das überhaupt gibt. Hiermit meine ich nicht, dass ein Benutzer keine Fehler machen kann, sondern seine naive Strategie, mit einem System umzugehen, immer die richtige ist und nur das System falsch auf seine Aktionen reagiert. Ziel eines Programms sollte es sein, dass der Benutzer es intuitiv richtig bedienen kann.

Alles umzusetzen was an Anregung und Kritik von den Versuchspersonen kommt, ist gar nicht möglich. Einige Äußerungen stehen einander sogar entgegen. So wird zum einen eine ausführlichere Ausgabe erwartet, die dem Benutzer klar sagt, was er jetzt zu tun hat, und zum anderen wird die Textmenge, die das System erzeugt, als zu umfangreich angesehen, so dass der Text nicht einmal gelesen wird. Daraus muss wohl abgeleitet werden, dass das System bessere und präzisere Formulierungen an vielen Stellen benötigt.

Eine Versuchsperson hat bei der Suche explizit angegeben, dass sie eine Telefonnummer sucht, beim Interview nachher sagte sie aber, dass sie gerne die Telefonnummer und Adresse als Ausgabe hätte, was die Standardeinstellung des Systems ist. Dieser Benutzer schien sich jedenfalls nicht im Klaren zu sein, dass in einem natürlichsprachlichen System jedes Wort auch gemäß seiner Bedeutung ausgewertet wird.

Die Dialogstrategie Nachfragen hauptsächlich danach auszuwählen, welche Frage das Ergebnis am stärksten verkleinert, wird von den Benutzern nicht angenommen. Das System muss an dieser Stelle sein Vorgehen wenigstens begründen, z.B. „Einen Herrn Meier gibt es nur in Berlin, nennen sie doch bitte die Straße in der er wohnt.“. Eine feste Nachfragehierarchie, in der auch Fragen gestellt werden, die nicht unbedingt notwendig sind, ist nicht wünschenswert. Zu wissen, welche Hierarchie von den Benutzern als sinnvoll empfunden wird, kann aber helfen, die Formulierungen der Nachfragen zu verbessern. Wie in dem gerade genannten Beispiel können dann die bekannten Informationen, von denen der Benutzer erwartet hätte, dass zuerst nach ihnen gefragt wird, in die Formulierung mit eingebaut werden.

Es ist aber davon abzuraten, zu viele vage Informationen in eine Nachfrage einzusetzten, da dann zum einen die Nachfrage länger wird und zum anderen die Antwort des Benutzers schlechter zu antizipieren ist. Ein Beispiel: „Einen Herrn Meier gibt es nur mit den Vornamen Klaus und Peter und er kann auch nur in Berlin oder Hamburg wohnen. Nennen sie doch bitte die Straße, in der er wohnt.“ In diesem Fall muss damit gerechnet werden, dass der Benutzer zu irgendeinem der angesprochenen Punkte was sagt, wenn er die Antwort nicht geben kann oder widersprechen will, z.B. „Ich suche aber Frau Anna Meier.“

Es ist ein Fehler in der Nachfragelogik enthalten, dass der Benutzer, wenn er eine Firma über die Gewerbebezeichnung sucht, nach dem Namen der gesuchten Firma gefragt werden kann. Diese Nachfrage sollte natürlich in einem solchen Dialogverlauf ausgeschlossen werden.

Die auch schon in Kapitel 2.4.5 vorgestellten Erkenntnisse von Strobel (1997) zur Dialogstrategie sind im Telefonauskunftsprogramm kaum aufgenommen worden, da die Quelle mir auch erst vorlag, nachdem dieser Programmteil schon implementiert war. Die darin vorgegebene Strategien haben aber wenig Einfluss auf die von den Versuchspersonen bemängelte fehlende Hierarchie der Nachfragen, da dort vor allem inhaltliche Fragen zur Datenlage gemeint waren, Strobels Theorie sich aber um formale Aspekte kümmert, d.h. es geht hier nicht um Fragen nach weiteren Informationen sondern z.B. darum, wie und wann zweifelhafte Informationen am besten bestätigt werden.

Als weitere Strategie für den Dialog ist aus der Evaluation zu entnehmen, dass die Benutzer stärker vom System geführt werden wollen. In jeder Dialogsituation müssen dem Anwender seine Handlungsoptionen aufgezeigt werden. Besonders nach Fehlermeldungen fehlt eine solche Unterstützung in dem Telefonauskunftssystem.

Dem Benutzer seine Handlungsoptionen vorzugeben, heißt aber auch, ihn in seinen Möglichkeiten einzuschränken. Für das System bringt dies sogar Vorteile mit sich, da die nächsten Handlungen des Benutzers besser vorauszusehen sind und eine Eingabe, selbst wenn sie nicht vollständig vom System verstanden wird, ausreichend Informationen enthalten kann, um erfolgreich weiterarbeiten zu können. Für den Benutzer bedeutet das hingegen, das er höchstwahrscheinlich nur noch tut, was das System ihm vorschlägt. Die Vorschläge dürfen aber nicht zu umfangreich sein, da Benutzer ja nicht zu lange Dialogzüge des Systems wünschen.

In diesem Prototypen ist keine Technik vorhanden, um die Einschränkung der natürlichen Sprache in diesem System aufzuzeigen. Am ehesten würde eine solche Komponente wohl benötigt, um den Benutzern aufzuzeigen, wie sie ihre Anfrage umformulieren müssen, damit der Computer sie versteht. Die zahlreichen Anmerkungen aus der Videokonfrontation zu diesem Thema bestätigen dies. Dies ist sicherlich einer der zentralen Punkte, an denen mit Dialogen eine deutliche Verbesserung erzielt werden können. Unbekannten Worten sollte wenn möglich die richtige Wortart zugeordnet und der restliche Satz so gut wie möglich weiterverarbeitet werden. Dann kann gezielt nach einem Synonym für das unbekannte Wort gefragt werden. In dem hier vorgestellten Programm ist dies leider sehr schwierig, da durch die Namen, Straßen, usw. schon viele unbekannte Worte verarbeitet werden müssen. Wenn nun ein wirklich unbekanntes Wort auftaucht, ist es nicht von den Namen zu unterscheiden.

Fragen die außerhalb der Domäne Telefonauskunft waren, gab es in der Evaluation nicht. In der Hinsicht haben wir es hier möglicherweise mit etwas zu tun, was Patrick und Whalen ein natürliches Thema nennen. Den Benutzern scheint klar zu sein, welche Informationen ein solches solches System zur Verfügung stellt. Eine Komponente, die dafür da ist, dem Benutzer die Einschränkungen der Domäne aufzuzeigen, wird somit wohl nicht zwingend benötigt. Möglicherweise liegt es aber nur an der Szenarios, die in der Evaluation eingesetzt wurden, dass die Benutzer keine Fragen außerhalb der Funktionalität des Programms gestellt haben.

Von der Funktionalität der Programms her scheint es nach der Evaluation zwingend notwendig geografische Relationen einzubauen. Einige Anwender benutzen Formulierungen wie „in der Region Osnabrück“ oder „Osnabrück und Umgebung“. Wahrscheinlich sind sich die Benutzer bei diesen Formulierungen nicht im Klaren, dass die ohne Weltwissen nicht aufzulösen sind. Ein Lösungsansatz für dieses Problem wurde schon in Kapitel 4.6 vorgestellt.

Für die in dieser Arbeit gewählte Domäne scheint ein kleines Lexikon ausreichend zu sein, da ein Großteil der Worte die eingegeben werden, Eigennamen oder ähnliches sind. In der Evaluation wurden von den Probanden nur 70 Worte verwendet die lexikalisch zu repräsentieren sind. Die Lexikongröße des Programms mit 140 Worten scheint somit ausreichend groß zu sein, auch wenn die 28 unbekannten Worte zeigen, dass die Auswahl der Worte im Lexikon noch überarbeitet und ergänzt werden muss.

Die Anzahl der eingegebenen Rechtschreibfehler ist wie erwartet gering (siehe auch Kapitel 2.5.2)

Dass der Benutzer seine Einstiegsfrage frei formulieren darf und dass der Computer sich selbst nicht als Computersystem bezeichnet, sind sicherlich keine optimalen Entscheidungen, da sie keine Vereinfachung der verwendeten Sprache hervorrufen, da sie weder Computertalk oder Konvergenz bei den Äußerungen des Gesprächspartners hervorrufen und auch nicht zu einfachen Benutzeräußerungen führen. Trotzdem wurden sie so gewählt, da ich es für interessant halte, die initiativen Züge den Benutzern zu überlassen und ihr Verhalten in dieser Situation zu beobachten. Durch die Ergebnisse der Evaluation kann nun möglicherweise ein besserer Einstieg in das Programm gefunden werden, bei dem der Benutzer zwar geführt werden sollte, es ihm aber auch möglich bleiben muss, den Dialog zu verkürzen.

Eine minimale Antwort auf eine Frage an eine maschinelle Schnittstelle ist vielleicht nicht wünschenswert, da sie in der natürlichen Kommunikation nicht in solcher Häufigkeit auftritt, aber sie sind bei den momentanen Möglichkeiten zur Verarbeitung der Sprache leider oft unvermeidlich, wie in gewisser Weise auch die Ergebnisse der Evaluation zeigen. Der Erfolg des Systems bei der Verarbeitung der Eingabe des Benutzers wäre deutlich höher, da z.B. die Antwort des Benutzers besser antizipiert werden könnte.

Benutzer zeigen die Tendenz, den Ergebnissen von NL-Schnittstellen nicht zu vertrauen. So berichtet Strobel (1997, S.15), dass Benutzer sich die Ergebnisse nach einem Experiment vom Versuchsleiter bestätigen lassen wollten. Durch ein teilweise unberechenbares Verhalten und dadurch, das Ergebnisse sich manchmal nicht reproduzieren ließen, verunsicherte das Telefonauskunftsprogramm die Probanden in der Evaluation zusätzlich. Schlechte Werte in der ergonomischen Qualität lassen sich möglicherweise auch auf diesen Aspekt zurückführen.

Zoeppritz (1983) berichtet, dass Benutzer von NL-Interfaces bei Fehlern die Schuld beim System und nicht bei sich selbst suchen, was eine wünschenswerte Tendenz ist. Benutzer haben für natürliche Sprache eine erheblich höhere Kompetenz und lassen sich, nach Zoeppritz, in dem Bereich auch nichts vom Computer vormachen.

In der hier vorgestellten Evaluation scheint es so, dass Benutzer schnell auf Schlagwortsuche und in einem Fall sogar auf Boolsche Suche zurückverfallen. Die höhere sprachliche Kompetenz nutzen sie nicht um eine alternative Formulierung zu finden und sich damit auf den „Gesprächspartner“ einzustellen, sondern es siegen die Erwartungen an maschinelle Schnittstellen und deren bekannte Fähigkeiten. Dies kann an den in diesem Falls sehr beschränkten sprachlichen Fähigkeiten liegen, aber das schnelle Ausweichen auf einfache Anfragen[36] zeigt auch wie sehr sich Benutzer an die existierenden Schnittstellen gewöhnt haben.

Wie auch schon die Tendenz zu Schlagwortanfragen zeigt, sind die Benutzer in vielen Bereichen schon sehr an einen „normalen“ Umgang mit Computer gewohnt. So versuchte auch nur eine Person während der Evaluation, sich die Ergebnisse ausgeben zu lassen, während das System noch Nachfragen stellte. Die anderen Probanden durchliefen immer erst alle Nachfragen, bis die Ergebnisse dann automatisch ausgegeben wurden. Eine Versuchsperson empfand es sogar als Schikane, dass ihr gesagt wurde, wie viele Ergebnisse das System

36 Mit der einen schon erwähnten Ausnahme.

gefunden hatte, ohne dass sie diese sofort sehen konnte. Deswegen gab es auch Verbesserungsvorschläge, einen Ausgabebutton einzubauen oder die Ergebnisse sofort auszugeben.

Das Unverständnis der Benutzer, dass die Ergebnisse nicht sofort ausgegeben wurden, kommt zum Teil aber auch durch den Zustand der Datenbank, die nicht besonders viele Einträge hatte. Wenn das System angibt, es habe 3 Einträge gefunden, versteht der Benutzer nicht, wieso nicht alle ausgegeben werden[37]. Wenn dort aber stehen würde das es 291 Ergebnisse gibt, wäre der Benutzer möglicherweise dankbar, dass das Programm ihm hilft, die Menge zu verkleinern.

Die Anregungen der Probanden zur Oberflächengestaltung und zum Feedback des Systems können größtenteils leicht umgesetzt werden. Die Kritik hätte vermieden werden können, wenn mehr Gestaltungsrichtlinien bei der Schnittstellengestaltung gleich umgesetzt worden wären. Aber es handelt sich bei dem Programm auch noch um einen Prototypen.

Dass die Versuchspersonen das Programm als innovativ und originell einstufen, zeigt durchaus, dass die Möglichkeiten von Dialogen bisher kaum genutzt wurden.

Die Dialogform an sich scheint den Versuchspersonen keinerlei Schwierigkeiten bereitet zu haben, es störte sie nicht, dass sie weiter befragt wurden, sondern die Probleme lagen mehr im Inhalt der Fragen und in der Steuerbarkeit des Dialogs.

Dass Probleme bei der Steuerbarkeit aufkamen, ist wohl eher darauf zurückzuführen, dass den Benutzern eine solche maschinelle Schnittstelle fremd ist und sie nicht auf die Idee kommen, Strategien aus dem zwischenmenschlichen Dialog zu übernehmen.

Die Generierung von Nachfragen durch ein Computerprogramm halte ich ohne umfassende Repräsentation der Domäne im System für nur schwer möglich. In Systemen wie USL oder Q&A, die ohne eine Beschränkung der Domäne auskommen, können zwar Anfragen als Verfeinerung der letzten Ergebnisse zugelassen werden. Selbst generierte Nachfragen können aber nur recht einfach sein, z.B. indem der Wert für eine Spalte erfragt wird. Wenn kodiert ist, welche Bedeutung die Werte in einer Spalte haben, sind vielleicht sogar elegante Formulierungen für die Fragen möglich.

37 Die Schwelle, ab wie vielen Ergebnissen eine Ausgabe erfolgt kann beliebig eingestellt werden. In der Evaluation war dieser Wert auf 1, da sich nicht viele Daten in der Datenbank befanden, aber wenigstens einige Dialogzüge durchlaufen werden sollten.

In dieser Arbeit wurde gezeigt, dass bedeutende Unterschiede zwischen geschriebener und gesprochener Sprache bestehen, die bei der Implementierung für das jeweilige Medium in Betracht gezogen werden müssen. Es ist davon abzuraten, eine Spracherkennung mit einem System für geschriebene Sprache zu kombinieren, ohne vorher Anpassungen am System vorzunehmen.

6.1 Ausblick

Viele für den Prototypen sinnvolle Erweiterungen und Verbesserungen wurden schon in Kapitel 4.6 genannt. Einige davon sind sicherlich auch für andere System mit anderen Anwendungsfeldern zu gebrauchen, auch wenn viele Erweiterungen nur im Bereich der Telefonauskunft Sinn manchen.

Durch die Ergebnisse der Evaluation sind zahlreiche weitere Fehler und auch Schwachstellen aufgezeigt worden. Viele der Fehler sind sicherlich leicht zu beseitigen, indem z.B. das Vokabular erweitert wird. Bei anderen Schwachstellen ist es aber nicht leicht, eine Lösung zu finden.

Wie auch schon in den Verbesserungen für das Programm (Kapitel 4.6) angesprochen, hat eine Benutzermodellierung einige Vorteile für das Programm, z.B. dass das Programm sich an die Ausdrucksweise des Benutzers anpassen kann, dass vom Benutzer gewünschte Ausgabeformate gespeichert werden oder dass deiktische Ausdrücke aufgelöst werden können.

Die Evaluation zeigt noch, dass ein natürlichsprachliches System möglichst fehlerfrei arbeiten muss, da die Benutzer sonst schnell das Vertrauen in das System verlieren und auf herkömmliche Anfragearten und Systeme ausweichen werden.

Für eine Weiterentwicklung des Programmes ist es auch von großem Vorteil, die Anforderungen einer richtigen Telefonauskunft zu protokollieren und auszuwerten. Zum einen kann damit festgestellt werden, welches sprachlichen Konstruktionen und Vokabeln in diese Domäne benötigt werden und zum anderen kann man prüfen, welche Funktionalität von einer Telefonauskunft, z.B. als Beratungssystem, erwartet wird.

Wie auch schon in den Kommentar von Zoeppritz (1995) zu USL zu erkennen ist, liegt meines Erachtens der aktuelle Schwerpunkt der kommerziellen NL-Entwicklung auf der

Weiterentwicklung der Spracherkennungstechniken. Die Verarbeitung der Eingabe scheint dabei nicht von so großer Bedeutung zu sein.

Auch das aktuelle System C@ll + Find und aktuelle Nachrichten[38] zeigen diesen Schwerpunkt der Forschung. Der Einsatz von natürlicher Sprache wird anscheinend besonders in über das Telefon zu bedienenden Programmen gesehen.

Kommerzielle Systeme, die Sprache nach syntaktischen und semantischen Verfahren auswerten, sind momentan nicht auf dem Markt vorhanden. Der Aufwand der Implementierung und des Betriebes solcher Systeme scheint zu groß zu sein.

Dialoge haben in der hier vorgeschlagenen Form noch keinen Einzug in die Schnittstellengestaltung gehalten. Aber wenn telefonisch zu bedienende Computersysteme erfolgreicher werden sollten, werden sich diese sicherlich auch von statischen vorher festgelegten Dialogen zu Systemen mit einer flexiblen Dialogkomponente weiterentwickeln.

Besonders in Bereich der sprachgesteuerten Systeme wird es notwendig werden, dass die Systeme selbst ihre Funktionalität erklären können. Da diese Systeme auf absehbare Zeit auch immer noch mit einer Teilmenge der natürlichen Sprache auskommen müssen, werden sie auch weiterhin sprachlichen Restriktionen unterliegen. Für diese beiden Probleme bieten Dialoge an einer Audioschnittstelle die beste Lösung.

Ob NL-Schnittstellen wirklich vom Benutzern angewendet werden, hängt am meisten von den individuellen Präferenzen der einzelnen Benutzer ab. Generell ist nur eine Akzeptanz zu erreichen, wenn eine NL-Schnittstelle auch wirklich sinnvoll eingesetzt werden kann. Bei Aufgaben, die mit anderen Methoden, wie der direkten Manipulation, einfacher zu lösen sind, werden natürlichsprachliche Schnittstellen hingegen keine Chance haben. Wie Arz et al. (1995) feststellen, müssen NL-Produkte entweder Lösungen für neue Anforderungen schaffen oder Vorteile gegenüber vorhandenen Programmen bieten.

Die Erweiterung natürlichsprachlicher Schnittstellen um eine Dialogkomponente verbessert diese Schnittstelle, so dass sie sich vielleicht, wenn das Dialogkonzept gut umgesetzt wird, zukünftig auf dem Markt besser behaupten kann. Vielleicht können Dialoge helfen, die Möglichkeiten von NL-Schnittstelle den Benutzern verständlicher zu machen und somit die Akzeptanz zu erhöhen.

38 http://www.heise.de/newsticker/data/jo-14.03.02-000/

7 Literaturverzeichnis

J. Arz, R. Flassig und E. Stegentritt. Natural Language Products in Restricted and Unrestricted Domains. In: G. Heyer und H. Haugeneder, Hrsg., Language Engineering: Essays in Theory and Practice of Applied Natural Language Computing. Seiten 211-224, Wiesbaden, 1995. Vieweg.

J.L. Austin. How to Do Things With Words. Oxford Univerity Press, Oxford, 1962

J. A. Banwart, S. I. Berrettoni. A Practical Approach to Testing a Natural Language System: Tools and Procedures. In: G. Heyer und H. Haugeneder, Hrsg., Language Engineering: Essays in Theory and Practice of Applied Natural Language Computing. Seiten 61-84, Wiesbaden, 1995. Vieweg.

J. R. Bergmann. Ethnomethodologische Konversationsanalyse. In: G. Fritz & F. Hundsnurscher, Hrsg., Handbuch der Dialoganalyse. Niedermeyer, Tübingen, 1994.

H.U. Block und S. Schachtl. What a Grammar of Spoken Dialogues has to Deal With. In: G. Heyer und H. Haugeneder, Hrsg., Language Engineering: Essays in Theory and Practice of Applied Natural Language Computing. Seiten 101-126, Wiesbaden, 1995. Vieweg.

P. Das-Gupta. Boolean Interpretation of Conjunctions for Document Retrieval. Journal of the American Society for Information Science, 38(4):245-254, 1987.

Duden, Band 4 – Die Grammatik. 6. Auflage, Brockhaus, Mannheim, 1998

S. Ervin-tripp. Childrens Verbal Turn-Taking. In: E. Ochs und B.B. Schieflin, Hrsg., Developmental Pragmatics, Seiten 391-414. Academic Press, New York, 1979.

C. A. Ferguson und C.E. DeBose. Simplified Registers, Broken Language, and Pidginization. In: A. Valdman, Hrsg., Pidgin ad Creole Linguistics, Seiten 99-125. Bloomington, London 1977.

D. Franck. Abtötungspartikel und Interaktionsmanagment. Tendneziöse Fragen. In: H. Weydt, Hrsg., Die Partikeln der Deutschen Sprache. Berlin, New York, de Gruyter, 1979.

B. Gattung. Der OPAC der Universitätsbibliothek Düsseldorf – Benutzerperspektive. In: E. Plassmann, Hrsg., Bibliotheken in Europa. 80. Deutscher Bibliothekartag in Saarbrücken 1990, No. 53; Sonderheft in Zeitschrift für Bibliothekswesen und Bibliografie, Seiten 103-108, Frankfurt a.M., 1991. Klostermann.

H. P. Grice. Studies in the Way of Words, Kapitel 'Logic and conversation', Seiten 22-57. Harvard University Press, Cambridge MA, 1989.

R. Grishman. Computational Linguistics: An Introduction. Cambridge et al., 1986.

A. G. Hauptmann und A. I. Rudnicky. Talking to computers: an empirical investigation. In: E. Hollnagel, G. Mancini, D. D. Woods, Hrsg., International Journal of Man-Machine Studies 28. Seiten 583-604. Academic Press, London, San Diego, 1988.

G. Hirst. Anaphora in Natural Language Understanding: A Survey. No. 119 in Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, New York, 1981.

L. Hitzenberger. Möglichkeiten bei der Anwendung phonologischer Verfahren bei der Namenssuche in Datenbanken. In P. Hellwig und H. Lehmann, Hrsg., Trends in der Linguistischen Datenverarbeitung, Seiten 71-85. Olms, Hildesheim 1986.

S. Herring. Interactional Coherence in CMC. Juni 1999.
http://www.ascusc.org/jcmc/vol4/issue4/herring.html

H. Kamp. A Theory of truth and semantic representation. In: J. A. G. Groenendijk, T. M. V. Janssen und M. B. J. Stokhof, Hrsg., Formal Methods in the Study of Language, No. 135 in Mathematical Centre Tracts, Seiten 277-322. Mathematical Centre, Amsterdam, 1981.

H. Kamp und U. Reyle. From Discourse to Logic. Kluwer Academic Publishers, Dordrecht, 1990.

S. Kannengießer. Korrespondenzen zwischen KI und Linguistik. In: K. v. Luck, Hrsg., Künstliche Intelligenz: 7. Frühjahrsschule, KIFS-89. Berlin et al., Seiten 270-282, 1989.

J. Krause. Mensch Maschine Interaktion in natürlicher Sprache. Niemeyer, Tübingen, 1982.

J.Krause. F&A und Q&A: Informationsabfrage in natürlicher Sprache, Was können die neuen Softwarepakete der Künstlichen Intelligenz leisten? In: R. Weingarten und R. Fiehler, Hrsg., Technisierte Kommunikation. Seiten 93-108, Opladen, 1988. Westdeutscher Verlag.

J. Krause. Computertalk and Multimodality. In: G. Heyer und H. Haugeneder, Hrsg., Language Engineering: Essays in Theory and Practice of Applied Natural Language Computing. Seiten 85-99, Wiesbaden, 1995. Vieweg.

C. Moranz. OPAC und OSIRIS. Vergleich einer strukturiert- und einer natürlichsprachlichen Eingabeschnittstelle für Literaturrecherche. Technical Report, Universität Osnabrück, Fachbereich Psychologie, 1998.

W.C. Ogden und C. Kaplan. The use of AND and OR in an natural language computer interface. In: Proceedings of the Human Factors Society - 30th annual meeting 1986, Seiten 829-833, Santa Monica, CA, USA, 1986. Human Factors Society.

W.C. Ogden und A. Sorknes. What do users say to their natural language interface? In: Proceedings of IFIP INTERACT '87: Human-Computer Interaction, 3. Human-Computer Interface Design: 3.3 Natural Language Dialogues, Seiten 561-566, Elsevier Science Publishers, North Holland, 1987.

A. S. Patrick und T. E. Whalen. Field testing a natural-language information System: usage characteristics and users' comments. In: S. Guildford, Interacting with Computers vol 4 no 2, Seiten 218-230. Butterworth-Heinemann, 1992.

A. S. Patrick, W. Jacques-Locmelis und T. E. Whalen. The Role of Previous Questions and Answers in Natural Language Dialogues with Computers. In. N. J. Norwood, International Journal of Human-Computer Interaction 5(2), Seiten 129-145. Ablex Pub. Corp., 1993.

F. C. Pereira und D. Warren. Definite Clause Grammars for Language Analysis – a Survey of the Formalism and a Comparison with Augmented Transition Networks. Artificial Intelligence 13, S. 231- 278. 1980 [Wiederabdruck in Grosz et al., S. 101-124, 1986]

Th. Peters. When smart People fail: an analysis of the transaction log of an online public access catalog. The journal of academic librarianship: articles, features, and book reviews for the academic library professionals, 15(5), Seiten 267-273, 1989.

N. Petreley. Down To The Wire. 1999.
http://iwsun3.infoworld.com/cgi-bin/displayArchive.pl?/99/13/o13-13.98.htm

H.-J. Postel. Die Kölner Phonetik. IBM-Nachrichten, 19:928-931, 1969.

R. Potthast. Einführung in die mathematischen Grundlagen der Künstlichen Intelligenz.
http://www.science-atlas.com/potthast/tutorial/ai/aiscript/aiscript.html
Göttingen, 2001.

M. Ronthaler. Dialogschnittstellen an Online-Informationssystemen: Notwendigkeit, Leistungsfähigkeit und Entwicklungsmöglichkeiten am Beispiel des OSIRIS-Systems. Osnabrück, 2000.

J. Searl. What is a Speech Act? In: S. Davis, Hrsg., Pragmatics: A Reader, Seiten 254-263. Oxford University Press, Oxford, 1975. reprint aus: M. Black (ed.): Philosophy in America, Allen und Unwin, New York, 1965.

B. Shneidermann. Direct Manipulation: A Step Beyond Programming Languages. Seiten 57-69, IEEE 8, 1983.

M. Strobel. Flexibilisierung und Beschränkung von Benutzeräußerungen innerhalb von telefonischen Mensch-Maschine Dialogen. Working Paper 28, IBM Deutschland Informationssysteme GmbH, Stuttgart, 1997.

Symantec Corporation. Benutzerhandbuch F&A. Version 1.1. München, 1986.

A. L. Thomas. Ellipsis: The Interplay of Sentence Structure and Context. Lingua, 47:43-68, 1979.

S. Varges. Parsing und Generierung in uniformen Architekturen. Düsseldorf, 1997.

W. A. Woods, R. J. Kaplan B. Nash-Webber. The Lunar Science Natural Language Information System: Final Report. BBN Report 2378. Bolt, Beranek and Newman, Inc., Cambridge, Mass., 1972.

W.A. Woods. A personal view on natural language understanding. SIGART Newsletters 61, Seiten 17-20, 1977.

M. Zoeppritz. Endbenutzersysteme mit 'natürlicher Sprache' und ihre Human Factors. In: H. Balzert, Hrsg., Software-Ergonomie – Tagung I/1983 des German Chapter of the ACM, Vol. 14: Berichte des German Chapter of the ACM, Seiten 391-410, Stuttgart, 4 1983. German Chapter of the ACM, Teubner.

M. Zoeppritz. 'Kommunikation' mit der Maschine. In: R. Weingarten und R. Fiehler, Hrsg., Technisierte Kommunikation. Seiten 109-121, Opladen, 1988. Westdeutscher Verlag.

M. Zoeppritz. Software Ergonomics of Natural Language Systems. In: G. Heyer und H. Haugeneder, Hrsg., Language Engineering: Essays in Theory and Practice of Applied Natural Language Computing. Seiten 33-60, Wiesbaden, 1995. Vieweg.

E. Zoltan, G. D. Weeks und W. R. Ford. Natural Language Communication with Computers: A Comparison of Voice and Keybord Input. In: G. Johannsen und J. E. Rijsdorp, Hrsg., Analysis, Design and of Man-Machine Systems, IFAC/IFIP/IFORS/IEA Conference, Seiten 255-260, Baden-Baden, 1982.